UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior
Departamento de Ingeniería de Sistemas y Automática
Laboratorio de Sistemas Inteligentes

Diseño de la arquitectura software de un microrobot para Eurobot

---

Proyecto fin de carrera

Ingeniería Informática

Autor: Erick Arranz Herrera
Tutor: José María Armingol Moreno

Leganés, julio de 2010

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Diseño de la arquitectura software de un microrobot para Eurobot

1 Introducción y objetivos

1.1 Introducción

Este proyecto nace en el año 2008, cuando mi compañero David Álvarez y yo, entramos a participar en el proyecto de Eurobot como trabajos dirigidos. En ese año se nos dio la oportunidad, gracias al nuestro tutor, Don José María Armingol, de conocer la robótica más de cerca.

En este proyecto se pudo conocer gente con más experiencia, y con muchos conocimientos e ilusión por la robótica.

Pronto se empieza a ver de lo interesante que podía ser el proyecto, y lo más importante, de lo que se podía aportar al mismo. Éramos los únicos ingenieros informáticos que habían entrado al proyecto, por este motivo se pudieron observar ciertos errores que existían, de mejoras que se podían hacer, y cómo no, se aprendieron cosas nuevas.

Gracias a este año, se descubrió lo importante que es trabajar junto con ingenieros de otras ramas, y lo que se pueden aportar unos a otros.

También estuvimos presentes en la copa de España, en Alcalá de Henares, en donde quedamos primeros, y pudimos asistir junto con todo el equipo a la final que se celebró en Heidelberg, Alemania.

El año siguiente, en 2009, empezamos en un nuevo Eurobot, esta vez como proyectistas, lo que nos dio pie a introducir nuevas ideas, dar otro punto de vista a la programación en la robótica.

Este proyecto empezó con mucha ilusión, gracias a lo que se aprendió en 2008, y las ideas nuevas que había que aportar para aportar al robot.

El año 2009 fue probablemente el que se aprendió de robótica. El hecho de ser proyectistas dio pie a poder investigar más a fondo el mundo de la robótica. También permitió aprender de los propios errores, aprender a cómo evitarlos, y a encontrar mejores soluciones.

Todo el trabajo en el año 2009 se dedicó a crear un software que fuese capaz de ser reutilizable. Hasta el ese año, todo el mundo que llegaba empezaba a programar desde cero, sin casi documentación. Esto se podía deber principalmente a la falta de gente que conociese más a fondo las distintas metodologías de programación. Normalmente el proyecto Eurobot había sido llevado a cabo por Ingenieros Electrónicos en su mayoría.

En el año 2010, volvimos a incorporarnos al equipo de desarrollo de Eurobot, siguiendo con el papel de programadores. Este se tenía la ventaja de toda la experiencia que habíamos adquirido, y toda la base software que ya estaba programada del año anterior.

Este año se pudo llevar a cabo ideas que habían surgido a lo largo del año 2009, como fruto de la experiencia, o de aprender de los propios errores. Se consiguió mejorar el software, y el hecho de haberlo utilizado para dos años distintos, demuestra la versatilidad del mismo.

El principal trabajo ha consistido principalmente en crear una base software reutilizable para cualquier Eurobot, o incluso para cualquier robot parecido. Y crear este documento como documentación de todo el proceso que se ha seguido.

1.2 Objetivos

El principal objetivo de este proyecto ha sido proporcionar a un microrobot una base de software que fuese reutilizable dentro del concurso Eurobot.

A lo largo de este documento se explica cómo ha sido desarrollada esta arquitectura a lo largo de los Eurobot en los años 2008 y 2009. Se indica en los distintos módulos en los que se ha descompuesto, y cómo pueden ser reutilizados. Esta parte se explica desde un punto de vista donde no se tiene en cuenta el concurso para el que fue diseñado, haciendo de esta manea hincapié en la reusabilidad del software.

También se explican las mejoras a nivel de hardware que han sido necesarias con el fin de poder desarrollar una mejor base software.

Dentro de la parte de la estrategia del robot, se explica cuál fue la estrategia general elegida en el año 2010, tanto la solución mecánica, de forma general, como la parte táctica del robot, donde se indican los distintos caminos que podía escoger el robot, y cuál se eligió y porqué.

La estrategia del Eurobot 2010 se muestra como ejemplo de uso del software para desarrollar un microrobot para Eurobot.

2 Estado del arte

2.1 Introducción a la robótica

¿Qué es la robótica? Los términos robótica y robot han tenido distintas definiciones a lo largo de la historia, vamos a señalar algunas:

"Robótica, Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales". (1).

El término robot se popularizó con el éxito de la obra RUR (Robots Universales Rossum), escrita por Karel Capek en 1920. En la traducción al inglés de dicha obra, la palabra checa robota, que significa trabajos forzados, fue traducida al inglés como robot. (2 págs. 26-27)

2.2 Clasificación de los robots

Los robots existentes se pueden clasificar, o bien por la generación en la que fueron construidos, o bien, por su arquitectura.

2.2.1 Según su cronología

1ª Generación.

Manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

2ª Generación.

Robots de aprendizaje. Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

3ª Generación.

Robots con control sensorizado. El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

4ª Generación.

Robots inteligentes. Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

2.2.2 Según su arquitectura

La arquitectura, es definida por el tipo de configuración general del robot, puede ser metamórfica. El concepto de metamorfismo, de reciente aparición, se ha introducido para incrementar la flexibilidad funcional de un robot a través del cambio de su configuración por el propio robot. El metamorfismo admite diversos niveles, desde los más elementales, cambio de herramienta o de efecto terminal, hasta los más complejos como el cambio o alteración de algunos de sus elementos o subsistemas estructurales. Los dispositivos y mecanismos que pueden agruparse bajo la denominación genérica del robot, tal como se ha indicado, son muy diversos y es por tanto difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los robots, con base en su arquitectura, se hace en los siguientes grupos: Poli-articulados, Móviles, Androides, Zoomórficos e Híbridos.

Poli-articulados

En este grupo están los robots de muy diversa forma y configuración cuya característica común es la de ser básicamente sedentarios, aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados, y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad". En este grupo se encuentran los manipuladores, los robots industriales, los robots cartesianos y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

Ilustración 2-1: Robot poli-articulado.

Móviles

Son robots con grandes capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

Ilustración 2-2: Robot móvil.

Androides

Son robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemática del ser humano. Actualmente los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación. Uno de los aspectos más complejos de estos robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del robot.

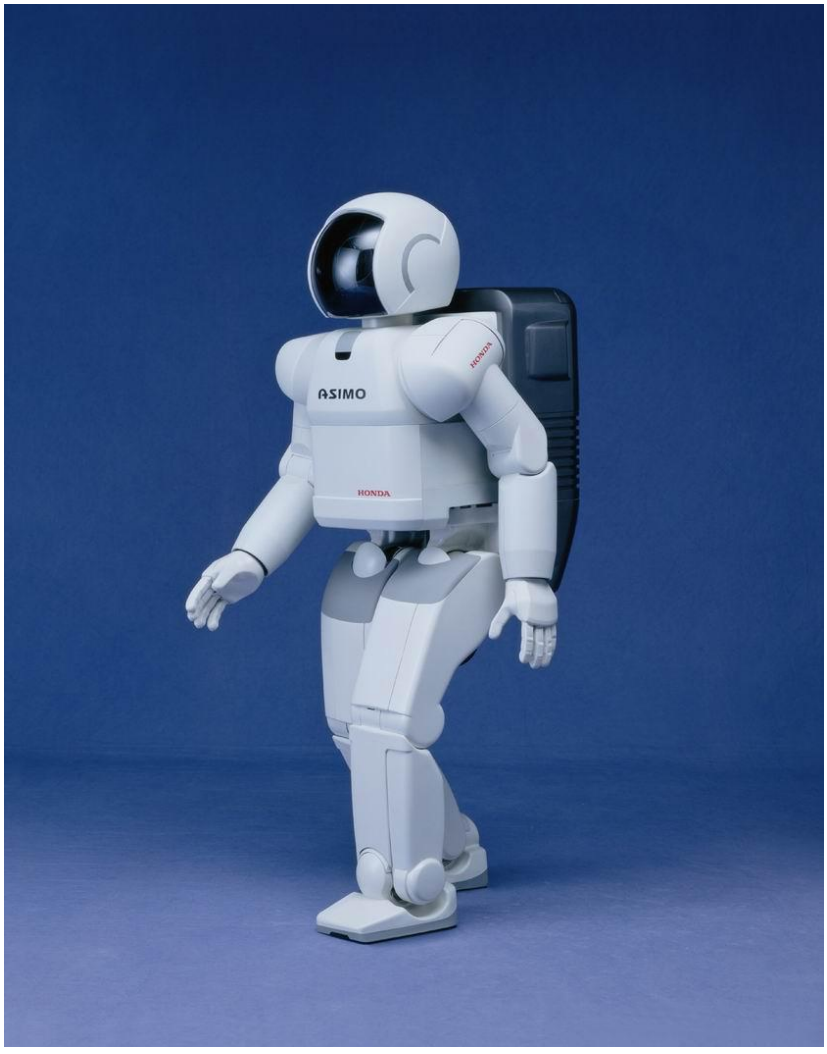

Ilustración 2-3: Robot androide ASIMO.

Zoomórficos

Los robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los robots zoomórficos no caminadores está muy poco evolucionado. Los experimentados efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. Los robots zoomórficos caminadores multípedos son muy numeroso y están siendo experimentados en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, piloteando o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

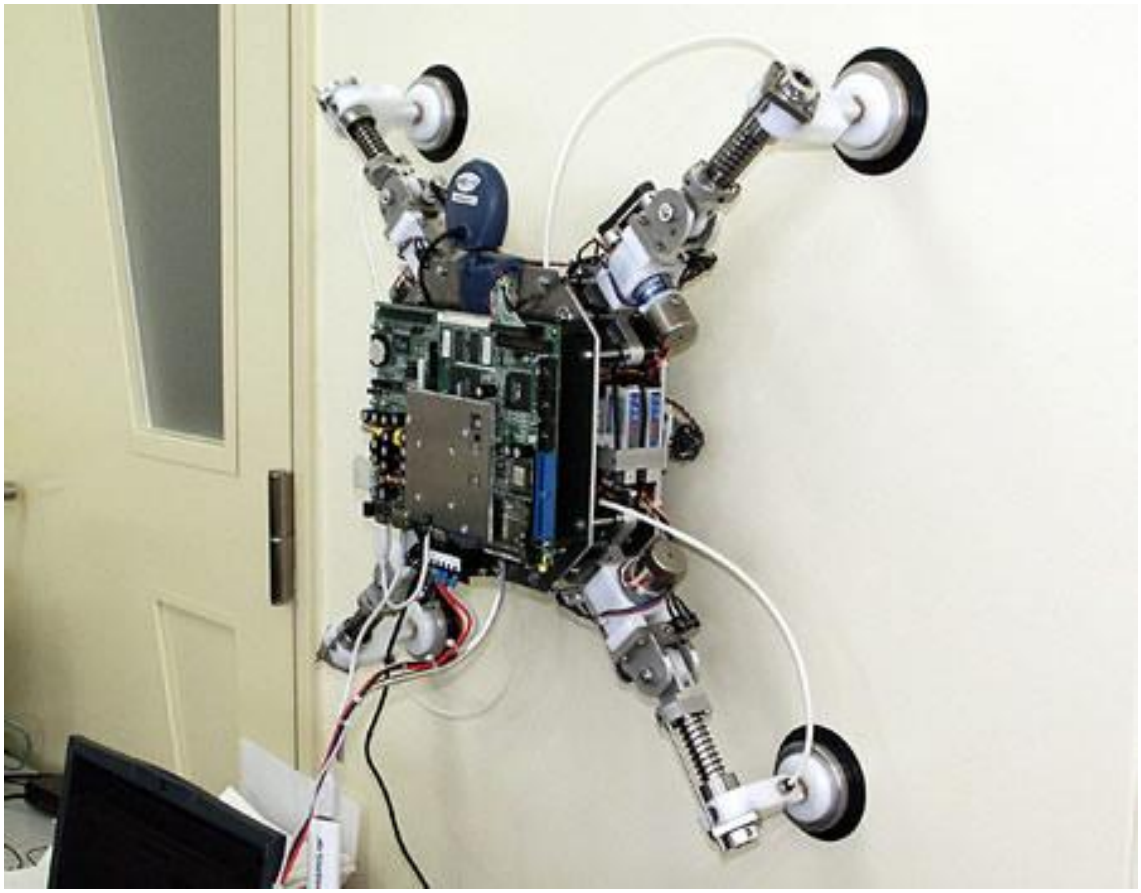

Ilustración 2-4: Robot araña.

Híbridos

Estos robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo uno de los atributos de los robots móviles y de los robots zoomórficos. De igual forma pueden considerarse híbridos algunos robots formados por la yuxtaposición de un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales. En parecida situación se encuentran algunos robots antropomorfos y que no pueden clasificarse ni como móviles ni como androides, tal es el caso de los robots personales.

Nanorobots

También llamado nanoagente, el término nanobot hace referencia a una máquina en la escala de los nanómetros.

La nanorobótica es la fabricación de máquinas, o robots, de dimensiones nanométricas. De una forma más específica, la nanorobótica se refiere a la todavía hipotética ingeniería nanotecnológica del diseño y construcción de robots. Otra definición, usada algunas veces, es la de una máquina capaz de operar de forma precisa con objetos de escala nanométrica.

Otra definición es un robot que permite la interacción con los objetos de precisión nanométrica, o puede manipular con resolución nanométrica. A raíz de esta definición, incluso un aparato de gran tamaño, como un microscopio de fuerza atómica, puede ser considerado un instrumento nanorobótico cuando se configura para realizar nanomanipulación. También nanorobots, robots a macroescala, o microrobots que pueden moverse con precisión nanométrica, puede también ser considerados como nanorobots.

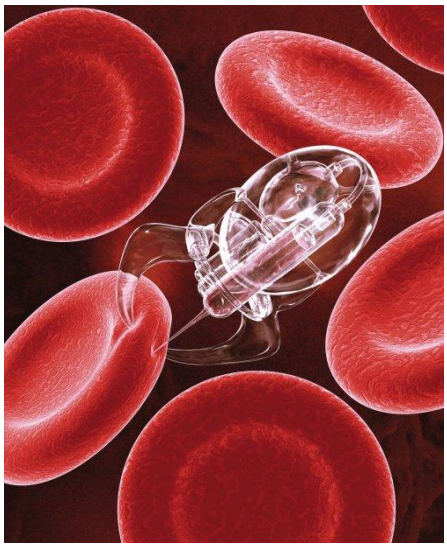

Ilustración 2-5: Ejemplo de un nanorobot.

2.3 Historia de los robots

La historia de los robots data por lo menos desde los antiguos mitos y leyendas. robots digitalmente controlados, robots industriales, y robots haciendo uso de la inteligencia artificial se han construido desde 1960.

Una breve cronología de la historia de los robots puede ser resumida en el siguiente diagrama:

Fecha	Importancia	Nombre del robot	Inventor
Siglo I a. C. y antes	Descripciones de más de 100 máquinas y autómatas, incluyendo un artefacto con fuego, un órgano de viento, una máquina operada mediante una moneda, una máquina de vapor, en Pneumatica y Automata de Herón de Alexandria	Autonoma	Ctesibius de Alexandria, Filón de Bizancio, Herón de Alexandria, y otros
1206	Primer robot humanoide programable	Barco con cuatro músicos robotizados	Al-Jazari
c. 1495	Diseño de un robot humanoide	Caballero mecánico	Leonardo da Vinci
1738	Pato mecánico capaz de comer, agitar sus alas y excretar.	Digesting Duck	Jacques de Vaucanson
1800s	Juguetes mecánicos japoneses que sirven té, disparan flechas y pintan.	Juguetes Karakuri	Hisashige Tanaka

1921	Aparece el primer autómata de ficción llamado "robot", aparece en R.U.R.	Rossum's Universal Robots	Karel Čapek
1930s	Se exhibe un robot humanoide en la World's Fairs entre los años 1939 y 1940	Elektro	Westinghouse Electric Corporation
1948	Exhibición de un robot con comportamiento biológico simple5	Elsie y Elmer	William Grey Walter
1956	Primer robot comercial, de la compañía Unimation fundada por George Devol y Joseph Engelberger, basada en una patente de Devol6	Unimate	George Devol
1961	Se instala el primer robot industrial	Unimate	George Devol
1963	Primer robot "palletizing"7	Palletizer	Fuji Yusoki Kogyo
1973	Primer robot con seis ejes electromecánicos	Famulus	KUKA Robot Group
1975	Brazo manipulador programable universal, un producto de Unimation	PUMA	Victor Scheinman
2000	Robot Humanoide capaz de desplazarse de forma bípeda e interactuar con las personas	ASIMO	Honda Motor Co. Ltd

Tabla 2-1: Cronología de la historia de la robótica

2.3.1 Robótica en la Antigüedad

Probablemente la ficción Ilíada ilustra el concepto de la robótica al afirmar que el dios Hefesto hizo hablar doncellas mecánicas de oro. Alrededor de 400 a. C., Arquitas de Tarento tiene fama de haber construido una paloma mecánica, posiblemente impulsada por vapor, capaz de volar. No sólo representa una de las primeras obras en el campo de la robótica, la paloma de madera fue también un estudio inicial de vuelo. Los filósofos (en particular, Aristóteles en el 322 a. C.) también han soñado con autómatas y herramientas capaces de trabajar de forma independiente de las personas como una idea de lograr la igualdad.

En la antigua China, un curioso número de autómatas se encuentran en el texto de Lie Zi, escrito en el siglo tercero antes de Cristo. Dentro de ella hay una descripción de un anterior encuentro entre tanto el rey Mu de Zhou (1023-957 aC) y un ingeniero mecánico conocido como Yan Shi, un "artífice". Este último presentó con orgullo al rey con una de tamaño natural, en forma de figura humana de su obra mecánica.

Los primeros relojes de agua o clepsidra, se agrupan a veces con el comienzo de la robótica. Era común intento de hacer que tales relojes automáticos o para decorar con complicados astrológica diseños (popular en el mundo oriental). De interés particular en China, estos relojes astrológicos dado lugar a complejas obras muy como Canción de Su torre del reloj en el 1088 AD, que aparece en movimiento maniquíes, entre otros dispositivos.

2.3.2 Robótica desde 1400 hasta 1800

El interés por los autómatas era, o bien no existente en la Europa medieval, o no registrados, sin embargo, encuentran su camino hacia los mundos imaginarios de la literatura medieval. Por ejemplo, el neerlandés Medio historia romana van Walewein ("El Romance de Walewein", siglo 13) describe pájaros mecánicos y los ángeles la producción del sonido por medio de sistemas de tuberías.

En 1495, Leonardo Da Vinci diseñó un autómata humanoide de una armadura (véase el robot de Leonardo) para entretener, pero no se sabe si el diseño se construyó alguna vez. Entre 1500 y 1800, se construyeron autómatas algunos capaces de hacer dibujos, volar, y tocar música; se construyeron también varias calculadoras mecánicas en este período, algunos de los famosos que más se Wilhelm Schickard "Cálculo de reloj", Blaise Pascal " Pascalina "y el" Leibniz escalonada del tambor " , por Gottfried Wilhelm Leibniz.

Ilustración 2-6: Robot de Leonardo da Vinci

Algunas de las famosas obras de la mayor parte del período fueron creados por Jacques de Vaucanson en 1737, incluyendo un autómata flautista, reproductor de pandereta, y su obra más famosa, " El Pato con aparato digestivo". El pato Vaucanson era capaz de imitar a un pato real agitando sus alas (más de 400 piezas se encontraban en cada una de las alas), comer el grano, digerirlo y defecarlo; el pato era accionado por medio de pesos.

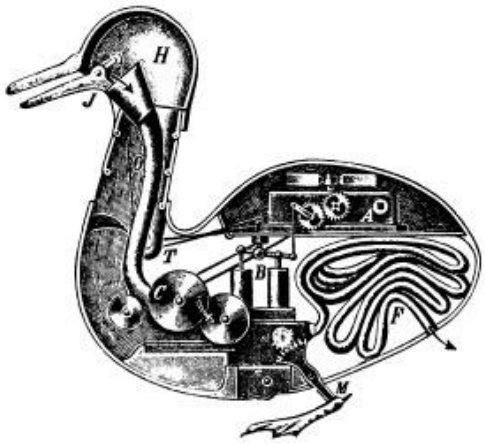

Ilustración 2-7: Representación errónea de cómo el Pato de Vaucanson funcionaba.

2.3.3 Robótica desde 1801 hasta 1900

Las mejoras en el tejido de la industria dio lugar a grandes cantidades de automatización, y la idea de máquinas programables se hizo popular con Charles Babbage y su Máquina Analítica. Babbage concibió su Máquina Analítica como un reemplazo para su inacabada máquina diferencial. Esta máquina más grande y compleja es capaz de realizar múltiples operaciones, y sería operado por tarjetas perforadas. La construcción de la máquina analítica nunca fue terminada, el trabajo se inició en 1833. Sin embargo, el trabajo de Ada Lovelace en el proyecto ha dado lugar a ser acreditada como la primera programadora de todos los tiempos.

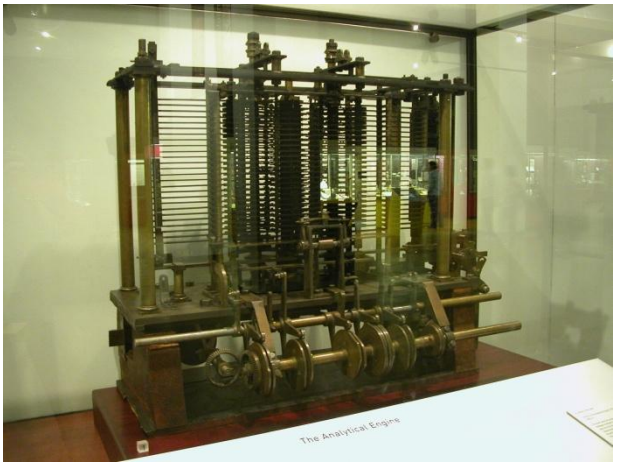

Ilustración 2-8: Prueba modelo de una parte de la Máquina Analítica, construido por Babbage, tal como se muestra en el Museo de la Ciencia (Londres)

George Boole inventó un nuevo tipo de lógica simbólica en 1847 instrumental para la creación de las computadoras y los robots.

2.3.4 Robótica desde 1901 hasta 1950.

La palabra robot fue popularizado por Checa autor Karel Capek en su juego 1921 RUR (Robots Universales de Rossum es). De acuerdo con Karel, su hermano Josef fue el verdadero inventor de la palabra "robot", la creación de la palabra de la palabra checa "robota", que significa servidumbre. La película de Fritz Lang, Metrópolis fue estrenada en 1926; María (personaje principal) fue el primer robot visto en una película.

Ilustración 2-9: Cartel de la película Metrópolis.

Muchos robots fueron construidos antes de los albores de los servomecanismos controlados por ordenador, a los fines de relaciones públicas de las grandes empresas. Electro apareció en el pabellón de Westinghouse en la Feria 1939 Mundial de Nueva York. Algunos fueron construidos en medio de tales reuniones públicas importantes, como Garco, mediante Garrett AiResearch en la década de 1950. Se trataba esencialmente de máquinas que pudieran realizar una acrobacias pocos, como los autómatas del siglo XVIII.

Vannevar Bush creó el primer analizador diferencial en el Instituto de Tecnología de Massachusetts (MIT). Conocido como el Analizador Diferencial, la computadora podía resolver ecuaciones diferenciales.

En el Reino Unido, la máquina Robinson fue diseñada para crackear los mensajes del Enigma. Robinson fue sustituido por Colossus, que fue construido en 1943 para decodificar los mensajes de FISH por el grupo británico Ultra, que fue diseñada por Tommy Flowers y fue de 100 a 1000 veces más rápido que Robinson. Fue la primera computadora completamente electrónica. Las máquinas de Bletchley se mantuvieron en secreto durante décadas, por lo que no aparecen en las historias de la computación por escrito hasta hace poco. Después de la guerra, Tommy Flowers se unió al equipo que construyó los primeros equipos de Manchester.

En Alemania, Konrad Zuse construyó la computadora programable completamente digital por primera vez en el mundo (Z3) en 1941; más tarde sería destruido en 1944. Zuse también fue conocido por la construcción de la primera computadora binaria desde 1936 hasta 1938, llamado el Z1, sino que también construyó el Z4, su máquina sólo para sobrevivir la Segunda Guerra Mundial.

La primera computadora programable de América se completó en 1944 por Howard Aiken y Grace Hopper. El Mark I (como se llamaba) publicó los cálculos para la Marina de los EE.UU. hasta 1959. ENIAC fue construida en 1946 y ganó fama por su fiabilidad, velocidad y versatilidad. John Presper Eckert y John W. Mauchly pasó 3 años construcción del ENIAC, que pesaba más de 60.000 libras.

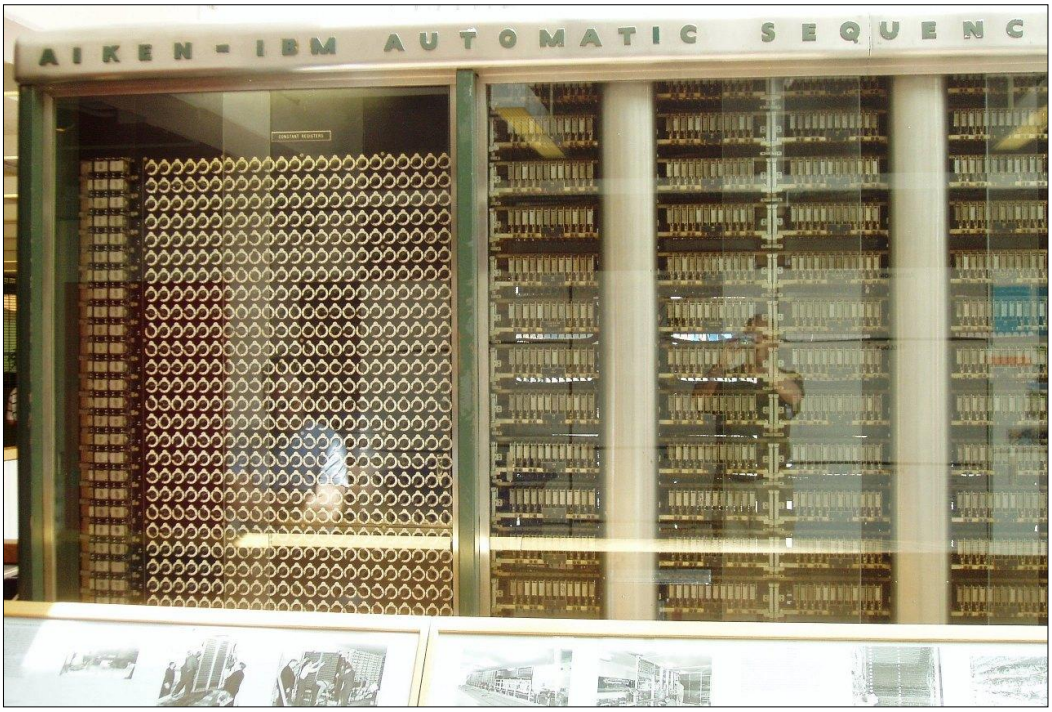

Ilustración 2-10: Parte del ordenador Harvard-IBM Mark 1, visto desde la izquierda.

La primera computadora digital de trabajo que se vendió fue de Zuse Z4 en Alemania, el totalmente electrónico EE.UU. BINAC se vendió doce meses antes, en septiembre de 1949, pero nunca funcionó de forma fiable en el sitio debido a un descuido del cliente en tránsito.

También en 1951, LEO comenzó a funcionar en el Reino Unido. Fue construido por el Lyon para su propio uso: este fue el primer ordenador digital electrónico programable por software del mundo para aplicaciones comerciales.

2.3.5 Robótica desde 1951 hasta el 2000.

A partir de 1950, las computadoras (y la robótica), empezó a aumentar rápidamente en complejidad y en número, así como la tecnología necesaria para producir los dispositivos, que ayudó enormemente a su desarrollo.

De 1951 a 1960

En 1952, la cadena de televisión CBS predijo correctamente la elección de Dwight D. Eisenhower como presidente con UNIVAC.

En 1952, IBM anunció su modelo 701, comercializados hacia el uso científico, que fue diseñado por Nathaniel Rochester, Stanislaw Ulam y el físico Paul Stein, convertida en MANIAC I (utilizado para resolver los cálculos implicados en la creación de la bomba de hidrógeno) para jugar un juego modificado de ajedrez en 1956, fue el primer equipo para vencer a un humano en un juego de ajedrez.

El término " inteligencia artificial se creó en una conferencia celebrada en el Dartmouth College en 1956.

De 1961 a 1970

Unimate, fue el primer robot industrial jamás creado. Comenzó a trabajar en la línea de montaje de General Motors en 1961, concebido en 1954 por George Devol y Joseph Engelberger durante el almuerzo. Unimate fue realizado por la empresa Unimation. Unimate es recordado como el primer robot industrial.

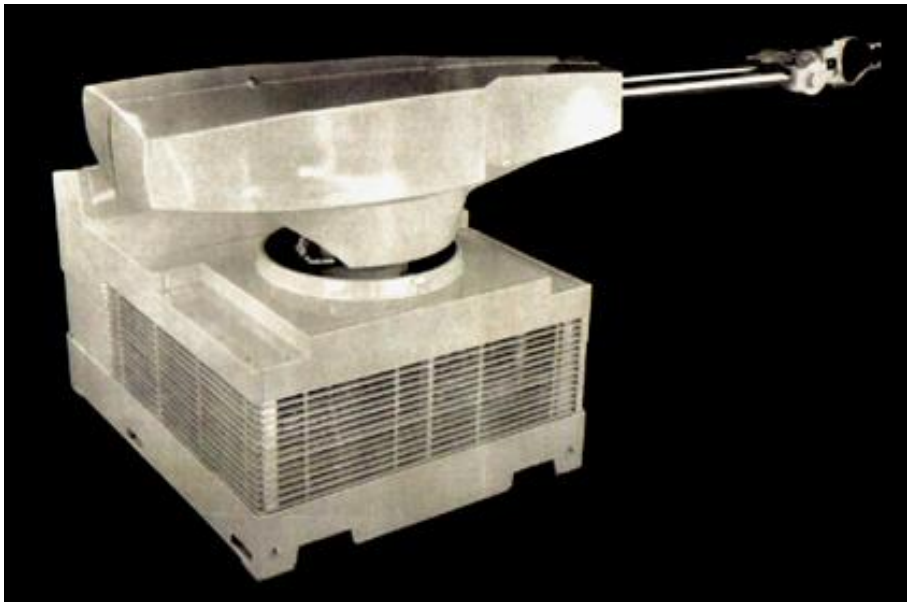

Ilustración 2-11: El robot Unimate original.

En 1962 John McCarthy fundó el Laboratorio de Inteligencia Artificial de Stanford en la Universidad de Stanford.

IBM anunció su sistema IBM 360 en 1964. El sistema fue anunciado como más potente, más rápido y con mayor capacidad que sus predecesores.

En 1965, Gordon Moore, co-fundador de Intel en 1968, desarrolla lo que se conoce como la Ley de Moore, la idea de que el número de componentes que pueden ser construidas en un chip se duplicará cada dos años.

Ese mismo año, el estudiante de doctorado Edward Feigenbaum , genetista y bioquímico Joshua Lederberg , y Bruce Buchanan (que tenía una licenciatura en filosofía) comenzar a trabajar en la DENDRAL , un sistema experto diseñado para trabajar en el campo de la química orgánica.

Feigenbaum también fundó el Proyecto de Programación Heurística en 1965, más tarde se convirtió en el Stanford Knowledge Systems Artificial Intelligence Laboratory.

La película 2001: Una odisea del espacio fue lanzado en 1968, la película cuenta con un lugar destacado HAL 9000 , una unidad de inteligencia artificial que controla malévola una nave espacial.

Marvin Minsky creó el brazo de Tentacle en 1968, el brazo estaba controlado por ordenador y sus 12 articulaciones fueron accionadas por sistemas hidráulicos.

El estudiante de Ingeniería Mecánica Víctor Scheinman, creó el brazo de Stanford en 1969, el brazo de Stanford es reconocido como el primer brazo robótico controlado por ordenador (las instrucciones Unimate se almacenaban en un tambor magnético).

El primer robot móvil capaz de razonar acerca de su entorno, Shakey fue construido en 1970 por la Stanford Research Institute. Shakey combinado de sensores múltiples entradas, incluyendo cámaras de televisión, láseres telemétricos, y "sensores de golpe" para navegar.

En el invierno de 1970, la Unión Soviética explora la superficie de la Luna con el vehículo lunar Lunokhod 1, la mecha del primer robot controlado por mando para aterrizar en otro planeta.

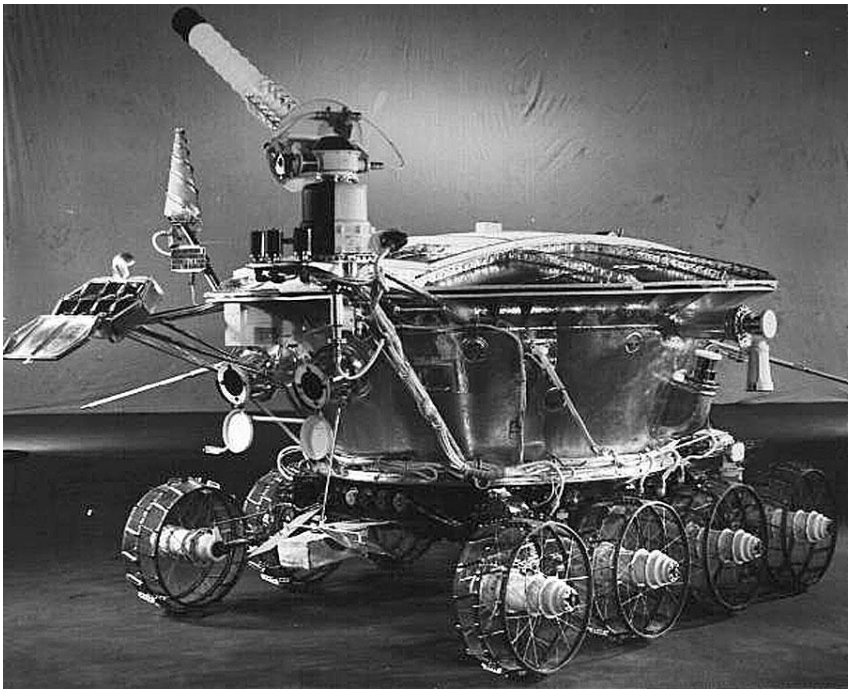

Ilustración 2-12: El robot soviético Lunokhod 1, creado para explorar la Luna.

De 1971 a 1980

El primer microprocesador, llamado el 4004 fue creado por Ted Hoff de Intel en 1971. El chip en sí mismo era más poderoso que ENIAC.

Freddy y II Freddy, ambos construidos en el Reino Unido, fueron robots capaces de ensamblar bloques de madera en un período de varias horas.

La empresa alemana KUKA construyó el primer robot industrial del mundo con seis ejes impulsados electromecánicamente, conocidos como fámulo.

En 1974, David Silver diseñó El Brazo de Plata, el Brazo de Plata era capaz de replicar los movimientos manos humanas con gran precisión

En 1975 más de 5.000 computadoras fueron vendidas en los Estados Unidos, y la primera computadora personal fue introducida.

La máquina de lectura Kurzweil (inventado por Raymond Kurzweil), destinada a ayudar a los ciegos, fue lanzada en 1976. Capaz de reconocer caracteres.

Basándose en estudios de objetos flexibles en la naturaleza (tales como trompas de elefante y las vértebras de las serpientes), Shigeo Hirose diseñó el agarre suave en 1976. La pinza era capaz de ajustarse al objeto que estaba agarrando.

Steve Jobs y Stephen Wozniak crearon el Apple Computer en 1977, y lanzó el Apple II.

Ilustración 2-14: Apple I. El primer ordenador de Apple.

La película Star Wars de George Lucas fue lanzado también en 1977. Star Wars contó con dos robots, un androide llamado C-3PO y R2-D2, los cuales llegado a ser unos icónos del mundo de la robótica.

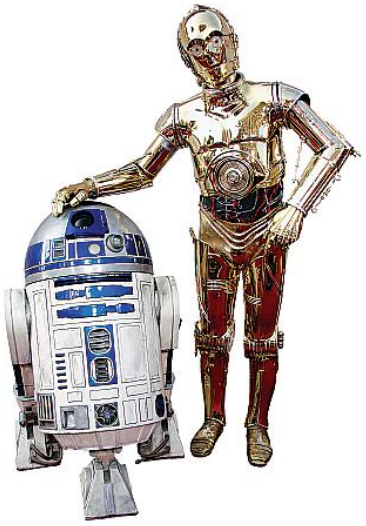

Ilustración 2-13: C-3PO y R2-D2 en la película Star Wars.

Las sondas Voyager 1 y 2 fueron lanzadas en 1977 para explorar el sistema solar. Los 30 años de edad sondas espaciales robóticas no les impiden seguir transmitiendo datos a la Tierra y se acercan a la heliopausa y el medio interestelar.

De 1981 a 1990

Takeo Kanade creó el primer "brazo de accionamiento directo" en 1981. El primero de su clase, los motores del brazo estaban contenidos dentro del propio robot.

IBM lanzó su primer ordenador personal (PC) en 1981, el nombre del equipo fue responsable de popularizar el término "computadora personal".

En 1984 el robot Wabot-2 fue mostrado, capaz de tocar el órgano, Wabot-2 tenía 10 dedos y dos pies. Wabot-2 fue capaz de leer una partitura de la música y acompañar a una persona.

En 1985, el acuerdo de licencia de Kawasaki Heavy Industries con Unimation se dio por terminado; Kawasaki comenzó a producir sus propios robots. Su primer robot fue puesto estrenado un año después.

En 1986, los ingresos de la inteligencia artificial fue alrededor de $ 1 mil millones de dólares EE.UU.

Los programas de ajedrez HiTech y Deep Thought derrotaron a maestros del ajedrez en 1989. Ambos fueron desarrollados por la Universidad Carnegie Mellon. Deep Thought allanó el camino para la Deep Blue.

En 1986, Honda comenzó sus investigaciones humanoides y programa de desarrollo para crear robots capaces de interactuar con éxito con los humanos.

Un robot hexápodo llamado Genghis fue mostrado por el MIT en 1989. Genghis se hizo famosa por ser rápida y barata debido a los métodos de construcción; Genghis usaba 4 microprocesadores, 22 sensores y 12 servomotores.

Ilustración 2-15: Ejemplo de robot hexápodo.

Rodney Brooks y Anita M. Flynn publicó "rápido, barato y fuera de control: un robot Invasión de la Sistema Solar". En el documento se propició la creación de pequeños robots baratos en mayor número para aumentar el tiempo de producción y disminuir la dificultad de poner en marcha robots en el espacio.

De 1991 a 2000

El robot biométrico RoboTuna fue construido por el estudiante de doctorado David Barrett en el Instituto de Tecnología de Massachusetts en 1996 para estudiar cómo los peces nadan en el agua. RoboTuna está diseñado para nadar y se asemejan a un atún de aleta azul.

Inventado por el Dr. John Adler, en 1994, el Cyberknife (un robot de radiocirugía estereotáctica) representa un método más rápido de realizar la cirugía con una precisión equivalente a la de los médicos humanos.

Ilustración 2-16: Robot Cyberknife.

Honda P2, el robot humanoide fue mostrado por primera vez en 1996. P2 era una parte del desarrollo del proyecto humanoide de Honda. P2 era más pequeño que sus predecesores y parecía ser más parecidos a los humanos en sus movimientos.

Tras esperar que sólo operase iete días, el Sojourner rover finalmente se apaga después de 83 días de operación en 1997. Este robot pequeño (sólo pesa 23 libras) realiza las operaciones semi-autónomas en la superficie de Marte como parte de la misión Mars Pathfinder. Equipado con un programa de evitación de obstáculos, Sojourner fue capaz de planificar rutas y navegación para estudiar la superficie del planeta. La capacidad de Sojourner navegar con pocos datos sobre su entorno y sus alrededores cercanos permite al robot reaccionar ante acontecimientos imprevistos y los objetos.

También en 1997, el programa jugador de ajedrez de IBM Deep Blue, venció el entonces Campeón Mundial de Ajedrez Garry Kasparov jugando en el nivel de "Gran Maestro". El súper ordenador fue una versión especializada de un framework elaborado por IBM, y fue capaz de procesar el doble de movimientos por segundo de los que había hecho durante el primer partido (que había perdido Deep Blue), supuestamente 200.000.000 movimientos por segundo. El evento fue transmitido en vivo por Internet y recibió más de 74 millones de visitas.

El robot humanoide P3 fue mostrado por Honda en 1998 como parte de la continuación del proyecto humanoide de la empresa.

Ilustración 2-17: Robot AIBO de Sony.

En 1999, Sony presentó el AIBO, un perro robótico capaz de interactuar con los humanos, los primeros modelos lanzados en Japón se agotaron en 20 minutos.

Honda reveló el resultado más avanzada de su proyecto de humanoides en el año 2000, llamado ASIMO. ASIMO es capaz de correr, caminar, comunicarse con los seres humanos y reconocimiento facial, voz y reconocimiento de posturas, e interactuar con su entorno.

Ilustración 2-18: Robot ASIMO, de Honda.

Sony también reveló su Sony Dream Robot, pequeños robots humanoides en el desarrollo para el entretenimiento.

En octubre de 2000, la Naciones Unidas estimó que había 742.500 robots industriales del mundo, con más de la mitad de los robots en Japón.

2001 hasta el presente

En abril de 2001, el Canadarm2 fue puesto en órbita y se adhirió a la Estación Espacial Internacional. El Canadarm2 es un brazo robótico más grande y capaz que la versión utilizada por el transbordador espacial, y es aclamado como "más inteligente".

También en abril, el vehículo aéreo no tripulado Global Hawk hizo el primer vuelo sin escalas autónomamente sobre el Océano Pacífico, desde la base aérea Edwards en California, hasta RAAF Edimburgo en el sur de Australia. El vuelo se realizó en 22 horas.

Ilustración 2-19: Robot Roomba de iRobot.

El popular Roomba, una aspiradora robótica, fue lanzado por primera vez en 2002 por la empresa iRobot.

En 2004, la Universidad de Cornell reveló un robot capaz de auto-replicarse, un conjunto de cubos capaz de conectar y desconectarse, el primer robot capaz de construir copias de sí mismo.

El 3 de enero y 24 de los Mars rovers, Spirit y Opportunity, aterrizan en la superficie de Marte. Lanzado en 2003, los dos robots recorrerán muchas veces la distancia de la prevista inicialmente, y están funcionando todavía.

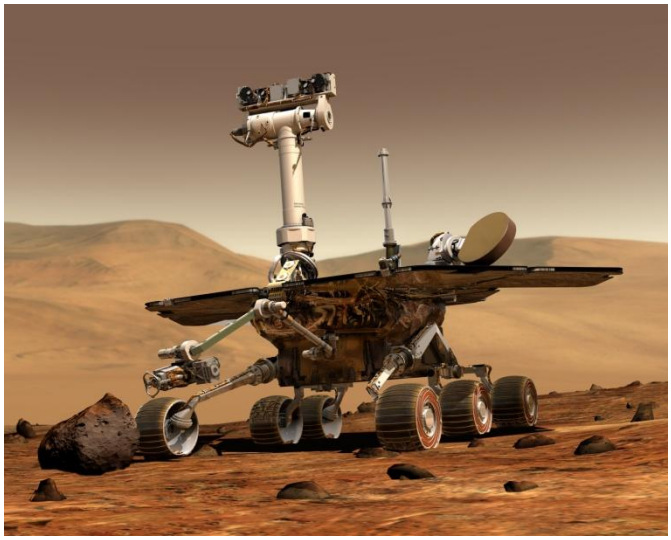

Ilustración 2-20: Robot explorador Spirit. Todavía operativo.

En 2005, Honda reveló una nueva versión de su robot ASIMO, actualizado con nuevos comportamientos y capacidades.

En 2006, la Universidad de Cornell reveló su "Estrella de mar" robot, un robot de 4 patas, capaz de modelarse a sí mismo y aprender a caminar después de de haber sido dañado.

En septiembre de 2007, Google anunció su Lunar X Priz. El Lunar X Prize ofrece 30 millones de dólares a la empresa privada primero que aterriza un rover en la Luna y envíe imágenes a la Tierra.

. En 2007, TOMY puso en marcha el robot de entretenimiento, i-SOBOT, que es un robot humanoide bípedo que pueda caminar como un ser humano y realiza patadas y puñetazos, y también algunos trucos divertidos y acciones especiales en "Modo de Acción Especial".

2.4 Concurso Eurobot

Eurobot es un concurso internacional de micro-robótica para aficionados. Está enfocado hacia equipos jóvenes, pudiendo ser clubes independientes o grupos de estudiantes con ganas de llevar a cabo un proyecto. Su finalidad es atraer a tantas personas como sea posible hacia la robótica y promover la práctica de esta ciencia.

La primera edición de Eurobot tuvo lugar en Francia en el año 1998 compitiendo en ella 5 países representados en total por 9 equipos. Aunque Eurobot tiene lugar en Europa, países de otros continentes pueden participar en el mismo. En 2004, 21 países, representados por 205 equipos se implicaron en este concurso científico y técnico, pasando por unas clasificaciones nacionales, y una final internacional. A lo largo de los años, la competición ha crecido tanto en importancia como en número de participantes, alcanzando en la edición de 1008 una participación de 400 equipos de 27 países, tanto europeos como de otras partes del mundo.

Cada país puede estar representado en la final por un máximo de tres equipos, por lo que se deben hacer etapas clasificatorias en cada país.

Cada año, este concurso propone un reto diferente a resolver, lo que hace de esta competición una de las más complicadas entre las existentes hoy en día en el campo de la micro-robótica, y a la vez uno de los más interesantes y divertidos, aunque siempre se mantienen ciertas normas todos los años.

Los distintos campeonatos de Eurobot a lo largo de la historia

Los campeonatos de Eurobot se han disputado en su mayoría en la ciudad de La Ferté-Bernad, en Francia. Sin embargo otros países también han organizado las finales, como Italia, Heidelberg, en Alemania, o Rapperswill-Jona, en Suiza.

2.4.1 Fútbol. 1998.

La competición era similar a un partido de fútbol jugado con más de una pelota. Sobre un terreno de juego totalmente plano en el que estaban distribuidas ocho pelotas de tenis, dos robots enfrentados tenían que marcar al rival el máximo número de tantos posibles. La final la disputaron nueve equipos de cinco países diferentes.

Ilustración 2-21: Eurobot 1998. Fútbol.

2.4.2 Ataque al castillo. 1999.

El campo se dividió en dos partes separadas un barranco, y unidas mediante dos puentes. En cada lado del campo se alzaba un castillo construido a base de cilindros de madera apilados de modo que formasen torres. El objetivo del juego era derribar el castillo del contrario ya fuese por contacto directo o mediante el uso de pelotas de tenis que se encontraban repartidas por el terreno de juego. Se presentaron ocho equipos de cinco países diferentes.

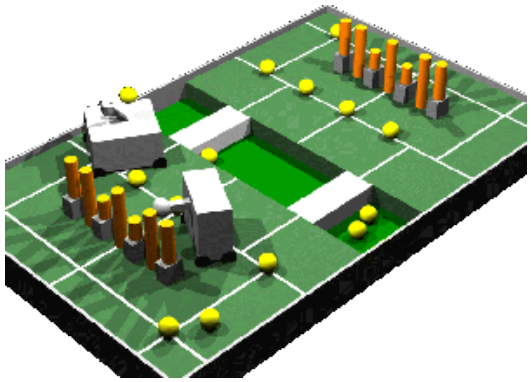

Ilustración 2-22: Eurobot 1999. Ataque al castillo.

2.4.3 Parque de atracciones. 2000.

Para la edición de ese año se diseñó un terreno de juego de superficie montañosa. Sobre él se dispusieron diez globos, cinco azules y cinco amarillos, y se dividió el campo en dos partes. Cada equipo debía reventar los globos del equipo contrario, estando prohibido el uso de proyectiles. Además la limitación impuesta al tamaño máximo de los robots, impedía reventarlos de lejos.

Se presentaron doce equipos de siete países diferentes.

Ilustración 2-23: Eurobot 2000. Parque de atracciones.

2.4.4 Odisea en el espacio. 2001.

En esta edición se dispuso un terreno de juego totalmente plano que representaba el espacio y sobre él se colocaron una serie de cilindro a modo de planetas. Los robots debían conquistar más planetas que el contrario depositando bandera de su color sobre los cilindros, reclamándolos de esta manera para su equipo. Se presentaron diecinueve equipos de doce países diferentes. Este año se hace una especial mención a la película de Stanley Kubrick 2001. Odisea en el espacio.

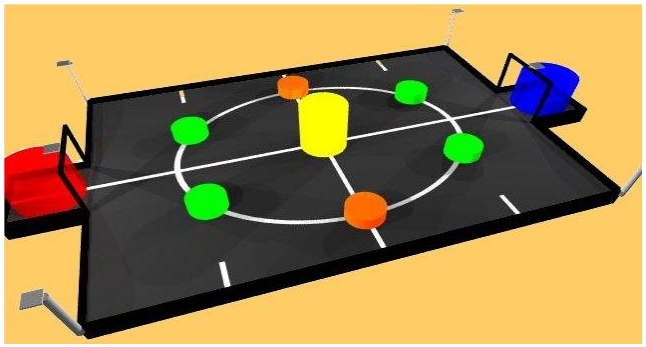

Ilustración 2-24: Eurobot 2001. Odisea en el espacio.

2.4.5 Billar aéreo. 2002.

En esta ocasión se dispuso un terreno de juego rectangular totalmente plano dotado de unos agujeros a modo de troneras de billar en cada esquina. Sobre el tablero se colocaban al azar ocho bolas rojas y cuatro negras siguiendo una simetría central. Cada robot comenzaba el partido en un extremo del terreno de juego u tenía que introducir las bolas negras en las troneras de su lado opuesto. Se presentaron veintisiete equipos de diecisiete países diferentes.

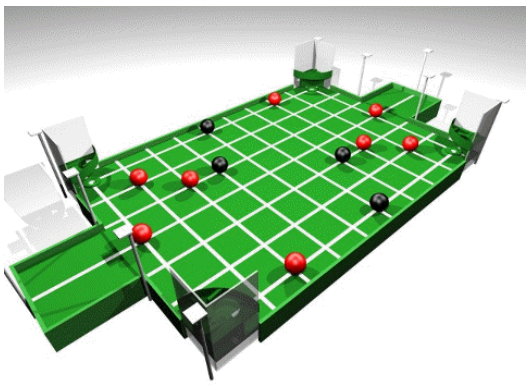

Ilustración 2-25: Eurobot 2002. Billar aéreo.

2.4.6 Cara o cruz. 2003.

En ese año, sobre un terreno de juego rectangular se colocaron una serie de discos de dos tipos diferentes, de doble color, y de color único. Los discos de doble color tenían una cara verde y la otra roja, y los de color único, podían ser verdes o rojos. A cada equipo se le asignaba un color, y al final del partido ganaba el robot que hubiese puesto cara arriba el mayor número de discos de su color. Se presentaros treinta y dos equipos de diecinueve países distintos.

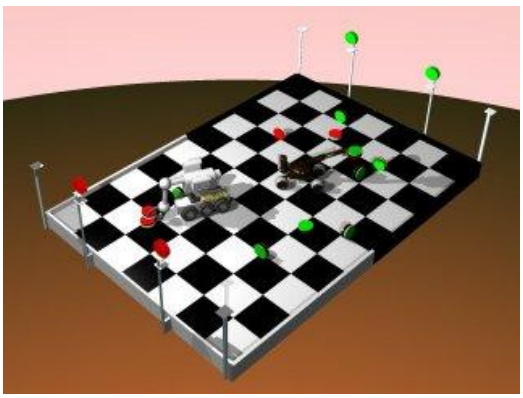

Ilustración 2-26: Eurobot 2003. Cara o cruz.

2.4.7 Rugby de cocos. 2004.

El terreno de juego diseñado para ese año era bastante peculiar: se trataba de un campo rectangular en el que se encontraban situadas una serie de palmeras
cargadas de cocos; además se colocaron dos porterías elevadas y una zona de ensayo en la superficie del terreno. El objetivo de la prueba de ese año era recoger los cocos que estaban distribuidos de manera aleatoria por el terreno de juego, tanto por el suelo como colgados de las palmeras, y marcar puntos lanzándolos a la portería o colocándolos en la zona de ensayo del lado del rival. Se presentaron cuarenta y un equipos de veintiún países distintos.

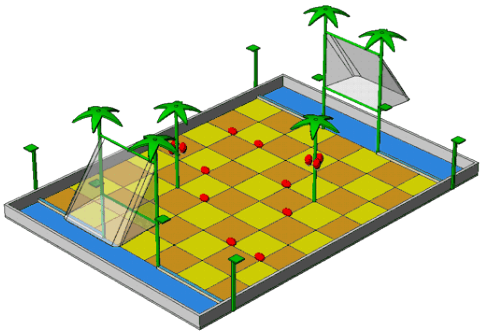

Ilustración 2-27: Eurobot 2004. Rubgy de cocos.

2.4.8 Juego de bolos. 2005.

Al igual que en la prueba de 1999 se dividió el terreno de juego en dos partes, separadas esta vez por un río, que se comunicaban entre sí por dos puentes de colocación aleatoria. En el centro del río existían dos carriles que permitían derribar los bolos del otro lado empujando una pelota. El objetivo de la prueba era derribar los bolos del contrario y proteger los propios. Se presentaron cincuenta equipos de veintidós países diferentes.

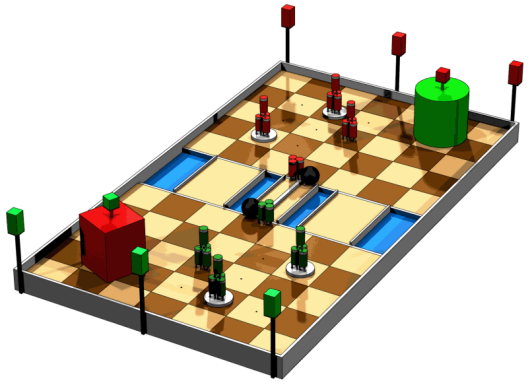

Ilustración 2-28: Eurobot 2005. Juego de bolos.

2.4.9 Golf divertido. 2006.

En esta ocasión el campo tiene multitud de hoyos donde los robots de cada equipo deben introducir pelotas de golf "blancas". Además existen pelotas "negras" que pueden introducirse en los hoyos del campo contrario, para evitar que el contrincante puntúe. Las pelotas de golf saltan al terreno de juego al cerrar un contacto situado en unos postes sobre terreno de juego. El tiempo total para la prueba era de un minuto y medio y cada equipo solo podía utilizar un robot.

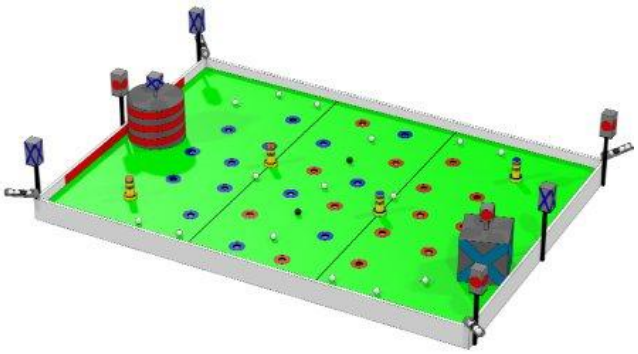

Ilustración 2-29: Eurobot 2006. Golf divertido.

2.4.10 Rally de reciclado. 2007.

Consistió en clasificar una serie de desechos (botellas de plástico, latas de refrescos y pilas) cada uno en su cesta correspondiente para su reciclado posterior. Ganó el robot que más desechos clasificó. El tiempo total para la prueba fue de un minuto y medio y cada equipo solo pudo utilizar un robot.

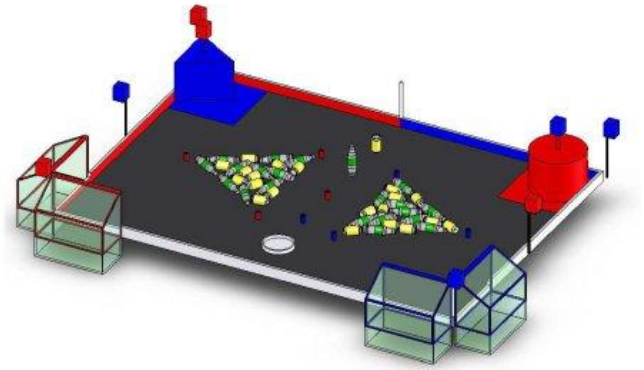

Ilustración 2-30: Eurobot 2007. Rally de reciclado.

2.4.11 Misión a Marte. 2008.

Los robots debían encontrar pruebas de vida y llevarlas a la Tierra para analizarlas. El robot que recogiera un número mayor de organismos, depositados correctamente en los contenedores sería el ganador.

La final de Eurobot 2008 se celebró del 21 al 25 de mayo de 2008 en Heidelberg, Alemania.

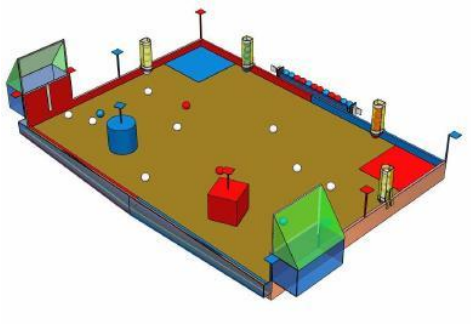

Ilustración 2-31: Eurobot 2008. Misión a Marte.

2.4.12 Templos de la Atlántida. 2009.

En este año la clave de la prueba era que el robot que construyera más templos y con sus columnas más altas sería declarado ganador. La competición consiste en dos robots de equipos distintos que se encargan de recolectar cilindros y piezas rectangulares del mismo color, piezas con las que se edificarán estructuras.

El concurso se celebró del 21 al 24 de Mayo en Ferté-Bernard (Francia).

Ilustración 2-32: Eurobot 2009. Templos de Atlántida.

3 Problema y solución Eurobot 2010

En este proyecto se pretende hablar sobre dos concursos de Eurobot, en los que se ha participado en el desarrollo del software.

Por esto se incluirán referencias hacia los dos concursos, correspondientes al año 2009 y 2010.

3.1 Problema Eurobot en el año 2010

El problema de este año consistía en recoger comida, repartida dentro del campo. El nombre que se le ha otorgado al concurso este año es "Feed The World", haciendo una analogía con robots que sean capaces de recoger comida. Se podría traducir como "Alimentando al mundo"

Ilustración 3-1: Logotipo de Eurobot 2010.

La comida puede ser o bien, tomates, mazorcas o naranjas.

Los tomates están representados por pelotas de malabares de color rojo y 10cm de diámetro dispuestas en el suelo del tablero, en posiciones ya determinadas.

Ilustración 3-2: Pelota que representa un tomate.

Las mazorcas están representadas por cilindros blancos o negros, colocados en puntos fijos del tablero.

Las mazorcas blancas están mínimamente enganchadas al tablero para poder ser extraídas sin dificultad, y las negras están atornilladas al tablero, siendo imposible su extracción. La elección de si una mazorca es blanca o negra es aleatoria, y se decide antes de cada partido sin dar tiempo a ningún equipo que modifique nada del robot, una vez elegida la disposición. Esto hace imposible programar una estrategia que sepa donde están las mazorcas blancas y negras de antemano. La única manera de saberlo es usando sensores de color, o bien una cámara que permita diferenciarlas.

Las naranjas son representadas por pelotas de malabares, de igual tamaño que los tomates, pero del doble de peso, 300gr, pero en este caso de color naranja.

Ilustración 3-3: Representación de un naranja.

Al igual que el resto de años, se sigue preservando ciertas normas comunes, como pueden ser las dimensiones del robot, del tablero, o el tiempo máximo de partido.

Este año han intentando hacer el partido más entretenido para el público, haciendo que los marcadores vayan mostrando en tiempo real los puntos que lleva cada equipo. Para ello han dotado a cada elemento del juego el mismo peso que puntos. De esta manera, colocando balanzas en las cestas de cada equipo, se puede medir la puntuación.

El partido es ganado por el equipo que obtenga más puntos, así de sencillo, y al contrario que otros años, la zona de dejar los puntos no está compartida, permitiendo a los robots hacer estrategias que sólo contemplen los puntos que haga cada uno, sin tener en cuenta lo que pueda realizar el robot oponente.

Según la disposición del tablero se pueden diferenciar dos zonas de puntos.

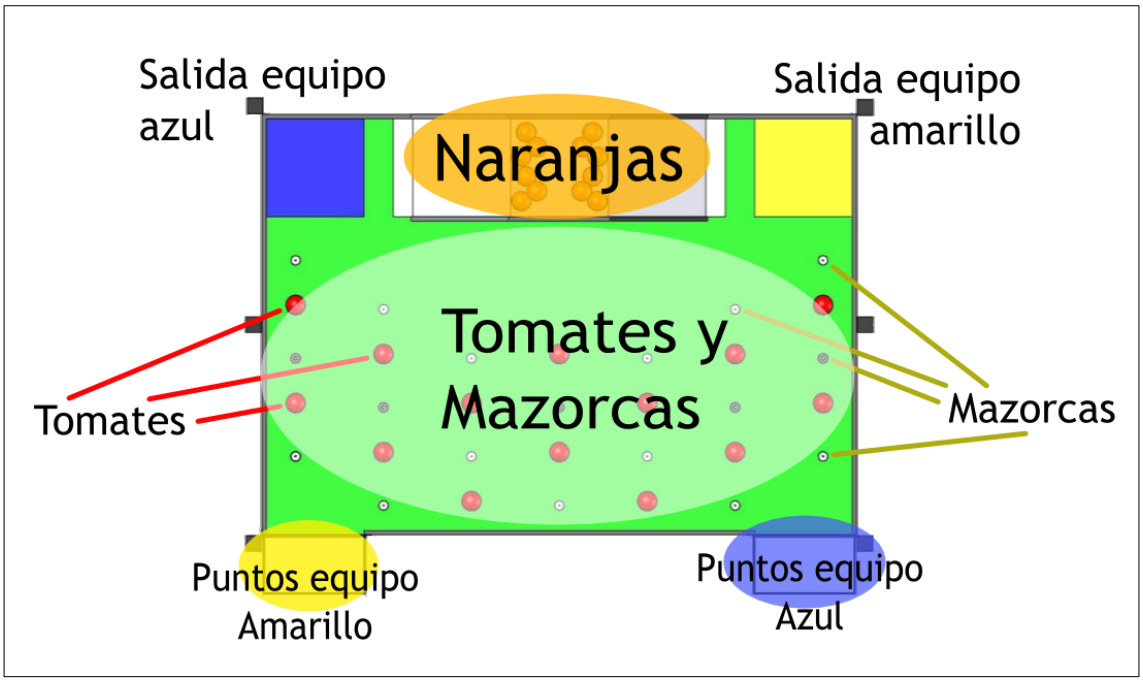

Ilustración 3-4: Disposición del campo de juego.

La primera es la zona de naranjas. El problema de esta zona, es que para acceder a ella, es necesario subir una rampa, no tiene demasiada inclinación, pero es necesario tenerlo en cuenta para el diseño del robot. Debido a que la inclinación del mismo repercute en el centro de gravedad, en el rozamiento con el suelo, y la fuerza que hacen los motores.

Por ejemplo si se diseña un robot alto, y con un punto de gravedad demasiado elevado. Es probable que al inclinarlo pierda estabilidad y se vuelque.

En la segunda zona se encuentran los tomates y las mazorcas. Esta zona hay que pasarla obligatoriamente si se quiere hacer algún punto, debido a que la zona de puntos está en la otra esquina de donde parte el robot.

3.2 Estrategia seleccionada

Por todos estos problemas señalados, se tomó la decisión de tener dos posibles caminos. El primero trata de recoger todos los tomates que se encuentren en el camino hasta la cesta de los puntos. Y como segunda opción es recoger naranjas.

El hecho de recoger o no naranjas, no interfiere con el resto de la estrategia. Esto es debido a que al recoger las naranjas se vuelve al mismo punto de inicio, y que el lugar de almacenamiento es distinto.

Por lo tanto se podrá activar la recogida de las naranjas de manera independiente.

3.2.1 Camino hacia la cesta

Existe un inconveniente este año, y es que llegar a la zona de puntos no es tan fácil como podía serlo en otros. Esto es debido a que existen las mazorcas que impiden un libre movimiento por el campo. Además la distancia entre mazorcas es demasiado pequeña, siendo ligeramente superior al ancho del propio robot.

Esto requiere que los movimientos del robot sean suficientemente precisos para no chocar con ninguna mazorca. Y para esto se hace uso de dos elementos. El primero los encoders para posicionar el robot, y saber en qué dirección ha de dirigirse. El segundo son los sensores de distancia. Con estos sensores se puede detectar la presencia de mazorcas en el lateral, o en el frente del robot, permitiendo de esta manera esquivarlos mucho más fácilmente.

También se debe fijar el camino que debe llevar el robot mientras esquiva las mazorcas. Hay que tener en cuenta que si se elige un camino para el color amarillo, por ejemplo, el del color azul será exactamente igual, pero simétrico.

En una primera instancia, si se analiza el campo detenidamente, se puede comprobar que existe un camino casi recto, en el que nos encontramos con tres tomates para puntuar.

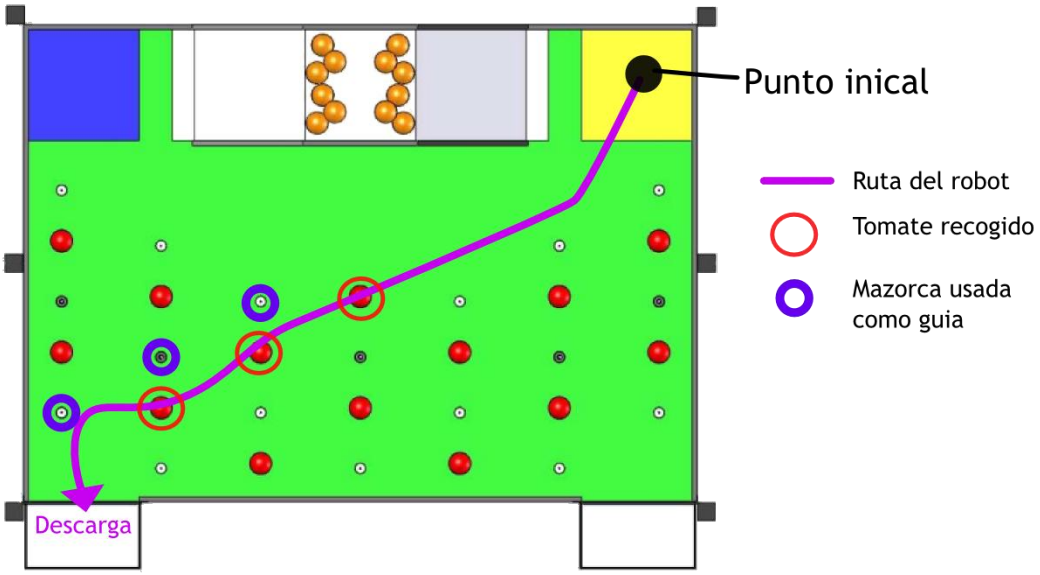

Ilustración 3-5: Primer camino posible

Este camino elegido tiene un único inconveniente, y es que el robot se encuentra durante mucho tiempo sin ninguna referencia, y esto puede ser un problema si los encoders no funcionan como deberían. Según la experiencia, la tasa de error de los encoders más los motores, puede generar un error suficiente para el robot pueda perderse en este camino, y no llegue al punto de referencia correcto.

Una vez visto que este posible camino puede traer distintos inconvenientes, se trató de estudiar otras alternativas. Viendo que los elementos que menos fallaban para el posicionamiento eran los sensores de distancia, se ha optado por una estrategia en donde se haga más uso de estos elementos.

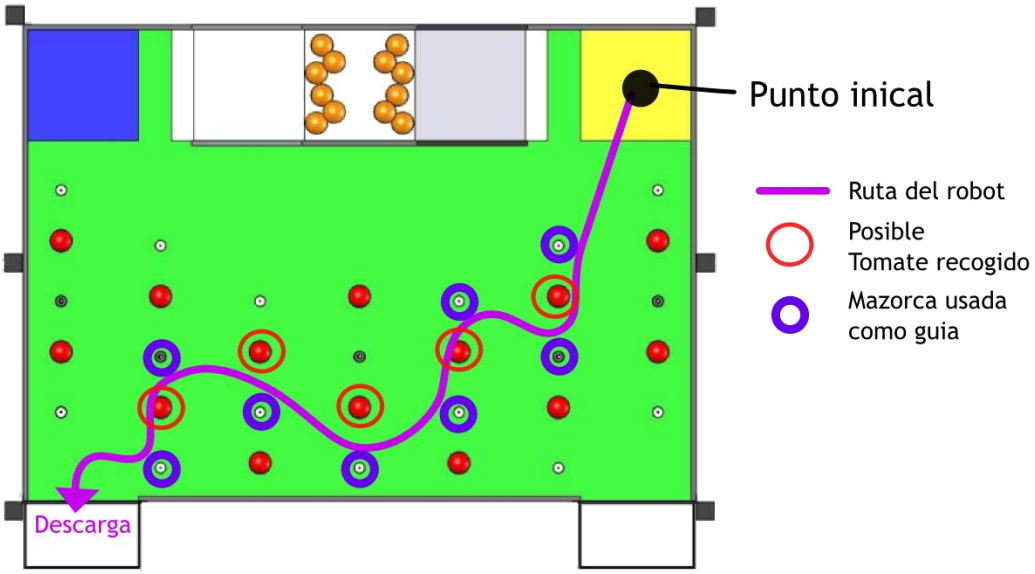

Ilustración 3-6: Camino alternativo para el robot.

Usando este camino se usan como guía más mazorcas, y debido a esto es más difícil perderse por el camino. Por lo tanto esta estrategia fue la elegida como **estrategia final**, y igualmente como **estrategia de homologación**.

3.2.2 Recogida de tomates

Otro problema que se plantea a la hora de recoger los tomates es que son muy pegadizos. Las pelotas de malabares oficiales que proporciona la organización tienen una textura que impide hacerlas rodar con sólo empujarlas. Si se intenta empujar con una superficie lisa, se podrá comprobar que es imposible hacer que los tomates se muevan. Igualmente pasa si queremos empujar un tomate con otro, se quedarán pegados y no se moverá ninguno de los dos.

Para evitar este problema se diseñó la zona de recogida del robot, de tal forma que los tomates nunca chocasen contra ninguna parte lisa del robot. Para esto se usan ruedas en la parte del robot, con la cual se quiere arrastrar un tomate.

También se diseñó para tener unas pequeñas palas que elevasen las pelotas, consiguiendo de esta forma que las pelotas nunca rodasen, tanto contra las paredes, como entre ellas.

Ilustración 3-7: Sistema de recogida de tomates.

Como se puede ver en la imagen, el sistema de recogida se basa en recibir las pelotas en la cavidad donde se almacenan, y una vez se encuentren en ella se pueden cerrar las palas de recogida, accionadas por dos servomotores, elevando las pelotas, y consiguiendo que no rueden.

Para descargar las pelotas, simplemente se abren las palas frontales mediante los servomotores, y se dejan caer sobre la cesta de los puntos.

3.3 Evasión del robot contrario

En las normas de Eurobot se indica claramente que se debe evitar chocar con el robot contrario en todo momento.

Dentro del proceso de homologación, esto se comprueba detalladamente, y en todo partido, si el robot aparenta no respetar esta norma, pude ser sometido a una nueva prueba de homologación para verificarlo.

La estrategia que se ha seguido durante estos últimos años es sencilla. Se utilizan unos sensores de distancia en la parte frontal del robot, a una altura en la que sólo se pueda detectar a un robot, y no otro tipo de elemento del campo. En el momento que estos sensores detecten a una distancia razonable algún objeto, el robot deberá pararse inmediatamente.

Debido a que el campo es muy complejo, este año no se puede optar por buscar un camino alternativo, en caso de que el contrario aparezca de frente. En la mayoría de los casos, no existe ningún camino alternativo si el robot contrario tapa alguna de las entradas. En todo caso casi siempre es mejor esperar a que se marche.

4 Esquema hardware del robot.

En este apartado se explicará cómo es el diseño mecánico y electrónico global del robot.

El robot se sustenta sobre dos motores, y sobre una base de aluminio endurecido. Las paredes, así como el techo, y las estructuras internas son de polimetacrilato de metilo.

El "cerebro" del robot, se encuentra en la placa Linux, que es la que recibe todas las señales que generan los sensores, y la que da las órdenes a los actuadores, que pueden ser los motores o los servos.

La electrónica de la parte actuadora está compuesta de dos placas. La primera de ellas es un micro-controlador, encargada de generar señales PWM para manejar servos y motores. La segunda placa, conocida como "Drivers", es la encarga de interpretar las órdenes que le llegarían a los motores, y transformarlos en voltaje y sentido de corriente, que hagan que los motores realicen las acciones que se desean.

Los actuadores que se encuentran en el robot son: los motores y los servomotores

En la parte sensiorial se compone de sensores GP2D12, GP2D120 y sensores de fin de carrera. Igualmente se usa una placa para interpretar las señales de estos elementos y transformarlas en información binaria.

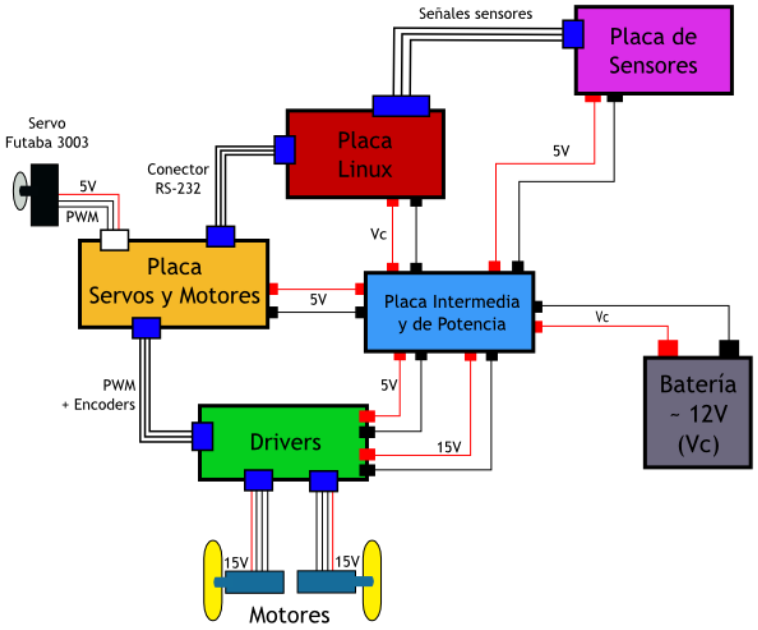

Ilustración 4-1: Esquema general del robot.

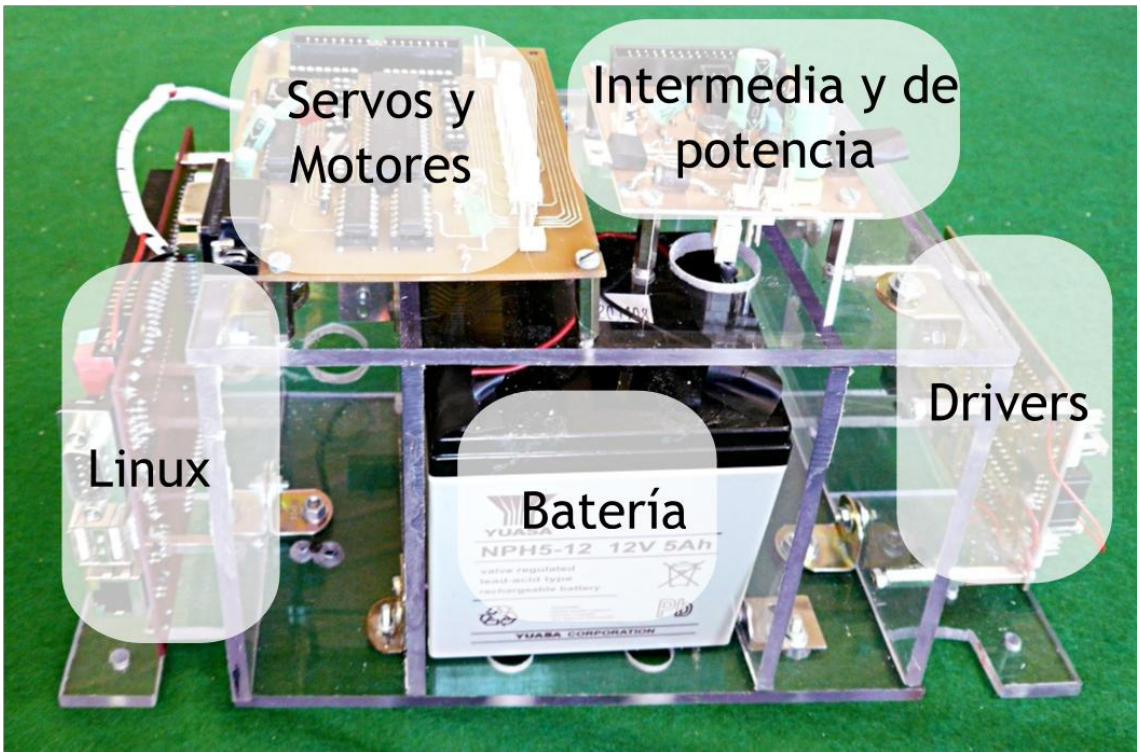

Ilustración 4-2: Esquema sobre el esqueleto del robot.

4.1 Placa principal Linux

Esta placa entra dentro de los ordenadores considerados como SBC (Single Board Computer), que sería como decir ordenador integrado en una única placa. Este tipo de dispositivos contienen todos los elementos necesarios para tener un ordenador operativo, con todos sus componentes integrados en la misma placa. Esto incluye el procesador, la memoria principal, el almacenamiento secundario.

También pueden incluir otro tipo de periféricos como pueden ser una tarjeta Ethernet, o controladores USB.

4.1.1 Componentes

En el caso del dispositivo TS-7350 las características principales son (3):

- CPU ARM9 a 200MHz
- 32MB SDRAM de memoria
- 5K LUT FPGA
- Conector flexible de 64-pin tipo PC/104
- Puerto Ethernet 10/100
- 2 puertos USB 2.0
- Ranura para tarjeta SD
- 3 puertos RS-232, 2 puertos TTL COM
- Cabeza de 40 pines con salidas y entradas de tipo ADC, SPI, I2C, DIO...
- Puede funcionar sin ventilador a una temperatura entre -40° y +70°C
- Entrada de voltaje desde 5 hasta 28VDC
- Arranca Linux 2.6 en aproximadamente 1 segundo.

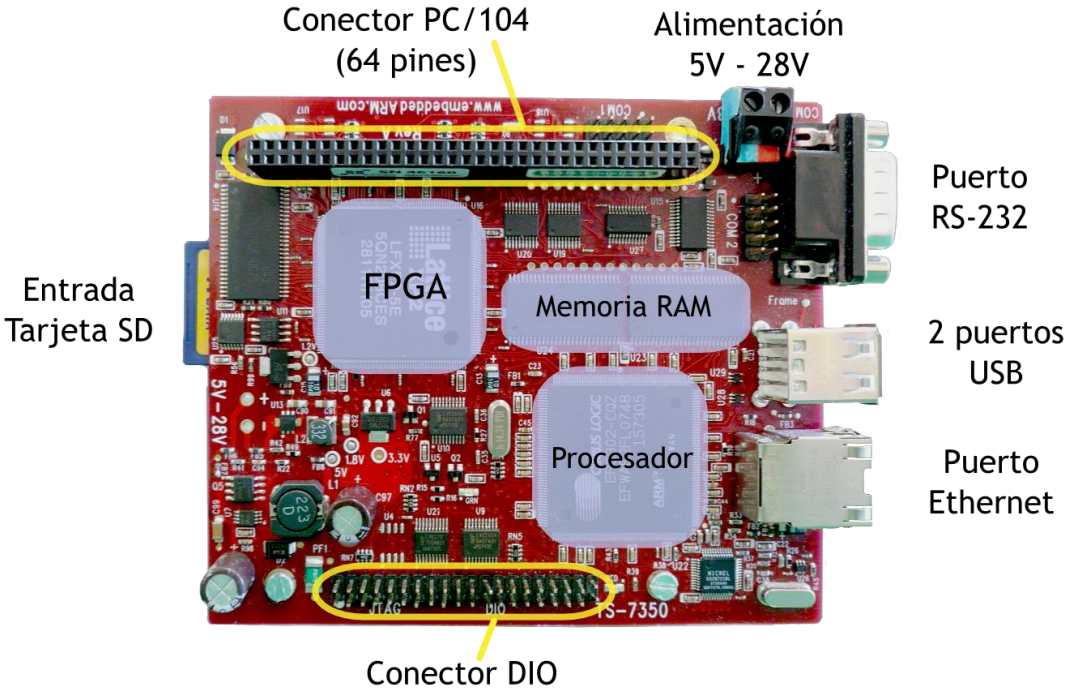

Ilustración 4-3: Esquema general de la placa TS-7350.

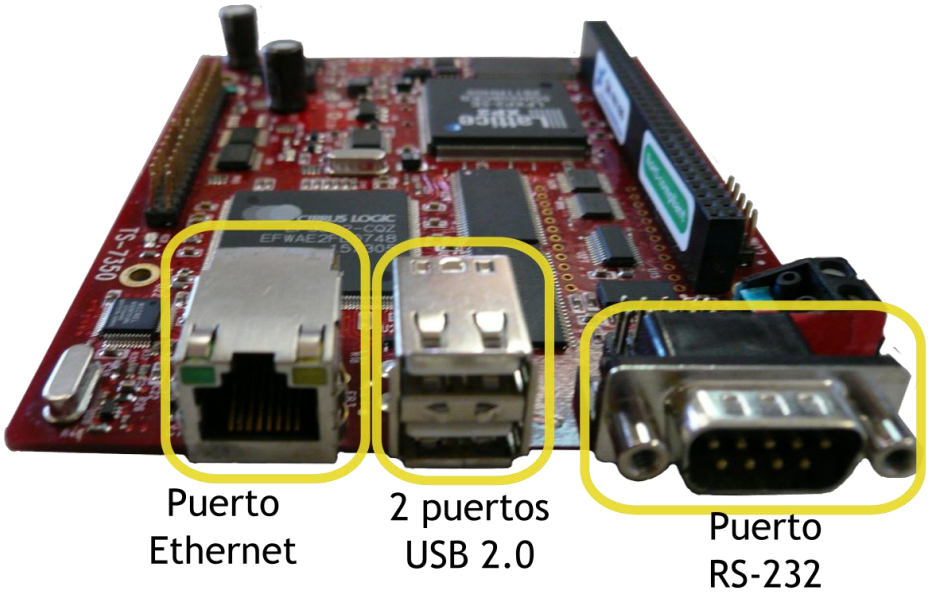

Ilustración 4-4: Vista frontal de la placa TS-7350.

4.1.2 Ventajas

Todas estas características hacen de esta placa un dispositivo muy útil para la robótica. El hecho de tener de serie puertos como el I2C, puertos RS-232, SPI, Ethernet o USB le permite comunicarse con cualquier tipo de dispositivo externo.

El hecho de incluir entradas y salidas, tanto analógicas como digitales, permiten el uso de elementos de muy bajo nivel, como puede ser un bumper, donde la única señal que se puede recibir es un 1 ó un 0.

También al tratarse de un ordenador donde se encuentra instalado Linux, permite hacer uso de las distintas herramientas que pueden encontrarse en un sistema operativo, como puede ser un servidor HTTP, un servidor FTP. O tener un sistema de ficheros completo, donde se puedan ir almacenando los distintos programas.

Por ejemplo el hecho de que tenga un puerto serie, nos permite comunicarnos con la placa de servos y motores sin problemas. Igualmente puede adaptarse cualquier dispositivo externo que use I2C, SPI, o entradas salidas digitales. También al disponer de 2 puerto USB, se pueden acoplar periféricos que en principio pueden no estar pensados para un sistema embebido, como ratones USB, o una cámara.

El hecho de contar con un conector Ethernet, hace que las comunicaciones con la placa sean sencillas, utilizando los protocolos TCP/IP para la comunicación.

Esta placa está pensada para entornos difíciles, donde exista voltajes de entrada variables, ruido en las señales que se reciben, y donde se requiere un consumo muy bajo.

4.2 Placa de servos y motores.

Esta es una placa desarrollada por integrantes de años anteriores de Eurobot.

Se trata de un circuito impreso que hace uso del micro-controlador DS89C450. Esta placa se construyó de forma modular, de manera que si un componente fallaba se podía reemplazar con gran facilidad, proporcionando de esta manera una gran eficacia en las labores de mantenimiento y puesta a punto del robot.

Gracias a este diseño modular y a que el micro-controlador seleccionado es compatible con otros modelos, esta placa se podrá utilizar indistintamente con dichos modelos de micro-controladores si en un futuro se requieren otras funcionalidades que el DS89C450 no pueda ofrecer.

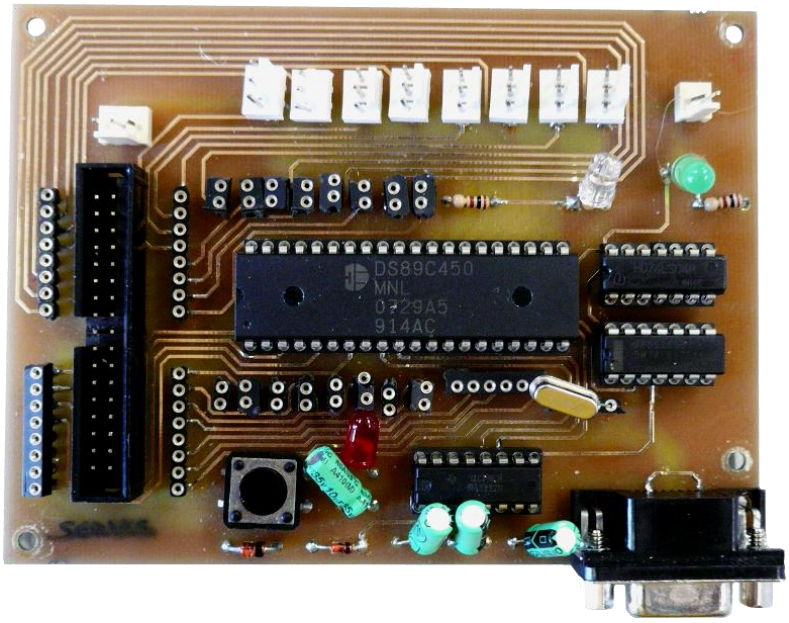

Ilustración 4-5: Placa de servos.

Cabe destacar que el diseño de estas placas se realizó pensando en posibles usos en futuras pruebas de Eurobot, por este motivo se intento realizar un diseño abierto y versátil que no limitase la posibilidades de desarrollo de futuros robots intentando aprovechar al máximo el rendimiento del micro-controlador. Por ello se facilitaron accesos a todos los puertos del micro-controlador tanto por conectores Molex como por un conector IDC de 34 pines, el cual además proporcionaba la alimentación necesaria para el funcionamiento de la placa de manera que esta podía operar conectando solamente un cable plano al conector IDC. (4 pág. 72).

Hasta la edición del 2008 usaban dos placas exactamente iguales, donde una controlaba los servos, y otra controlaba los motores y los encoders. La comunicación y sincronización se realizaba mediante entradas y salidas digitales.

A partir del año 2009 se paso a usar únicamente una placa que controlase todos los servos, motores, y leyese los encoders de los motores.

Estas placas han sido reutilizadas por un motivo fundamental, y es que desde la placa de Linux existen operaciones que no se pueden hacer, como es por ejemplo, la generación de señales PWM, o contadores de pulsos, necesarios para leer los encoders de los motores.

Entonces en este momento la placa Linux pasa a contener toda la lógica de control, que incluye las estrategias, algoritmos de parada y evasión del contrario.

La placa que antes se encarga de controlar el robot, ahora pasa a un segundo plano, y se convierte en un dispositivo que simplemente recibe órdenes.

4.3 Motores

Los motores escogidos son unos motores de corriente continua Bernio modelo MR 615 30 Q con una reductora de 1/16. La combinación del motor y la reductora nos ofrece unas características de par y velocidad más que suficientes para cubrir las necesidades del proyecto, con un consumo de energía aceptable.

La dirección de giro se controla a través de la polaridad de sus terminales, aligual que todos los motores de corriente continua, así que, si se invierte la polaridad también lo hará el sentido de giro. La velocidad en este tipo de motores viene determinada directamente por el nivel de tensión aplicado en sus terminales, de manera que cuanto mayor sea el voltaje aplicado, mayor será la velocidad de giro. De esta manera se puede controlar fácilmente el sentido y la velocidad del motor.

Un aspecto a tener en cuenta en cualquier motor, es el pico de corriente de arranque, que de no ser bien controlado puede dañar e incluso inutilizar el motor. En este caso, el pico de corriente de arranque es de 5,4 A y no se puede mantener más de dos segundos seguidos.

Ilustración 4-6: Motores Bernio.

4.4 Placa de Drivers

Para la alimentación de estos motores se ha utilizado el driver LMD18200T que proporciona hasta 3 A y añade la potencia necesaria a la señal PWM generada por programación.

Esta placa es la encargada de invertir la polaridad de los motores en caso de que se desee cambiar el sentido.

5 Arquitectura software del robot

El software principal encargado de controlar el robot se encuentra dentro de la placa Linux, en donde se tienen que llevar a cabo la mayoría de las operaciones importantes, y las decisiones que debe tomar el robot.

Al tratarse de un programa para el sistema operativo Linux, se puede desarrollar un programa más complejo de lo que se haría con un micro-controlador, por ejemplo. Se puede descomponer en distintos módulos, crear librerías reutilizables, y distintos programas de ayuda.

El lenguaje utilizado ha sido C. Esto se debe fundamentalmente a que se requiere en muchos casos un acceso a bajo nivel a distintos elementos, como por ejemplo, acceso al puerto serie, o acceso directo a memoria para utilizar las entradas y las salidas digitales.

También es necesario utilizar C para poder desarrollar un programa lo más cercano a un programa en tiempo real que se pueda. Y es que con otros lenguajes, esto es algo directamente impensable.

5.1 Descomposición modular del software.

Con el fin de tener un software lo más reutilizable y desacoplado posible, se ha separado en distintos módulos, cada uno con una tarea diferente.

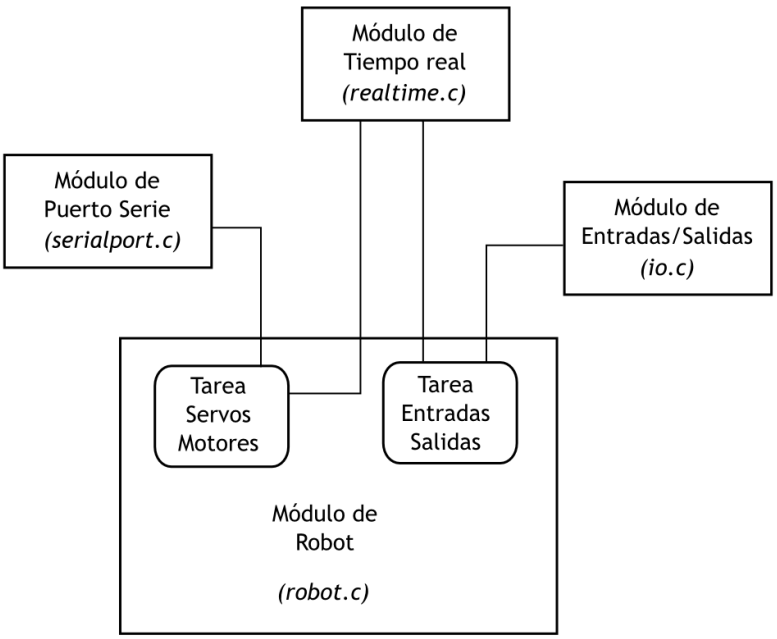

Ilustración 5-1: Descomposición en módulos del sofware.

5.1.1 Módulo del puerto serie

Dentro de este módulo se encuentran las funciones necesarias para utilizar correctamente el puerto serie, que es necesario para la comunicación con la placa de servos y motores.

La primera función que tiene este módulo es la correcta configuración del puerto serie, para que la comunicación sea correcta.

Los puertos serie tiene muchos tipos de configuración, según se quiera utilizar paridad, el número de bits de parada, la velocidad en baudios, el control de flujo, etcétera. En concreto para la conexión con la placa de servos y motores, la configuración debe ser la siguiente:

Velocidad en bits/segundo	57600 bits/seg
Número de bits de datos	8 bits
Bits de parada	1 bit
Control de paridad	Ninguno
Control de flujo Hardware	Ninguno
Control de flujo Software	Ninguno

Tabla 5-1: Parámetros de configuración del puerto serie.

La segunda función que desempeña este módulo es la de implementar el protocolo establecido con la placa de servos y motores. La descripción del protocolo que se usa, está explicado más adelante en el apartado "5.5 Protocolo de comunicación por el puerto serie"

Este módulo se encarga de comprobar que todas las órdenes que se manden lleguen correctamente, y se establecen tiempos máximos de espera para detectar caídas, o desconexiones.

Las funciones que podemos encontrar en este módulo son las siguientes.

```
int serial_open(char *device_name);

int serial_reset(int device);

int serial_servo(int device, unsigned char num_servo, unsigned char
position);

int serial_motor(int device, int left_speed, int right_speed);

int serial_motor_factor(int device, int left_factor, int right_factor);
```

```
int serial_encoder_left(int device);

int serial_encoder_right(int device);

int serial_motor_steps(int device, unsigned char steps);

int serial_close(int device);
```

5.1.2 Módulo de tiempo real

Dentro de este módulo se engloban todas las tareas de tiempo real.

Cuando se hace una aplicación es importante conocer que existen detalles del funcionamiento de los sistemas operativos que escapan al control de cualquier aplicación. Es por esto que en los sistemas tipo POSIX se provee de ciertas funciones que ayudan a evitarlas.

Un claro ejemplo es el desalojo de la memoria. El sistema operativo, puede verse en un punto desbordado y quedarse sin memoria para las aplicaciones. La manera de combatir esto es pasar ciertas páginas de memoria al disco duro. Esto puede ocurrir en cualquier momento y sin previo aviso. Esto puede ser letal para una aplicación en tiempo real. Debido a que el tiempo que se puede tardar en pasar desde la memoria hasta el disco duro, y viceversa puede ser entorno a 500ms, ¡y todo este tiempo la aplicación se queda bloqueada!

La manera de evitar esto es usando la función "mlock". Que impide que la memoria sea desalojada.

Se incluyen funciones para crear hilos con prioridad en tiempo real, para crear mutex que eviten la inversión de prioridad, usando o bien, herencia de prioridad, o techo de prioridad.

También se incluye código de tratamiento de señales, como puede ser un Ctrl+C.

Las funciones externas, que se encuentran definidas en el archivo *realtime.c* son las siguientes:

```
int realtime_init(int priority);
void realtime_set_timeout(int seconds);
```

```
rt_mutex_t *realtime_new_mutex(void);

rt_thread_t *realtime_start_thread(int (*routine)(void), int interval, int
prority);

void realtime_thread_finish(rt_thread_t *thread);

int realtime_run(int (*routine)(void), int interval);

void realtime_stop_all(void);

void realtime_set_finish_function(void (*routine)(void));
```

5.1.3 Módulo de Entradas/Salidas

El tratamiento de las entradas y las salidas dentro de la placa TS-7350 es un poco especial. Hay que tener en cuenta que, al contrario que con un micro-controlador, en nuestra aplicación no tenemos acceso a toda la memoria lógica del procesador, únicamente a la virtual del programa. Leer y escribir en ciertas posiciones de memoria, es la manera de leer y escribir en entradas y salidas digitales, pero como ya hemos dicho esto no es posible en un sistema opertivo multitarea.

Por suerte existen funciones que nos permiten acceder directamente a la memoria, en concreto haciendo uso del dispositivo /dev/mem dentro del sistema operativo.

En este módulo se encuentran todas las funciones que nos evitan tener que acordarnos en qué posición de memoria se encontraban las salidas digitales.

Las funciones que se encuentran definidas en "io.h" son las siguientes.

```
int io_init();

void io_read_a_data(unsigned char *inputs);

void io_read_b_data(unsigned char *inputs);

void io_read_dio(unsigned char *inputs);
```

5.2 Tareas de tiempo real

Dentro del software encargado de controlar el robot, existe una capa de abstracción llamada módulo "robot". Este módulo contiene una serie de funciones que permiten abstraernos de lo que hace internamente. Pero para poder entenderlo hay que conocer cuáles son las tareas que se están ejecutando por debajo.

5.2.1 Tarea de control de motores y servos

Esta tarea se está ejecutando continuamente, y es la encargada de comunicarse mediante el puerto serie con la placa de servos y motores.

Debido a que el protocolo está diseñado siguiendo un paradigma de parada y espera. Esta tarea se ejecuta continuamente, enviando las órdenes a la placa, y a la vez leyendo los encoders, y actualizando variables internas. De esta forma se puede medir la velocidad real a la que se está moviendo el robot.

Esta es la tarea que más prioridad tiene, y no tiene un periodo de ejecución. Se ejecuta lo más rápido posible. La velocidad a la que se ejecuta viene dada por la velocidad que tarden en enviarse y recibirse los mensajes por el puerto serie.

5.2.2 Tarea de entradas y salidas digitales

Esta tarea es la encargada de leer las entradas y salidas digitales de la placa Linux. El cometido principal de que esta tarea esté ejecutándose en paralelo es poder detectar pequeñas activaciones de sensores que con un ciclo de tarea muy alto serían difíciles de detectar.

Por ejemplo, si tenemos una tarea principal que se ejecuta cada 500ms, pero un sensor se activa durante un periodo de 10ms, este pequeño pico nunca será detectado.

Es por este motivo que esta tarea tiene un periodo de ejecución muy bajo, de apenas 5 milisegundos.

En caso de que se detecte activa una entrada digital, esta quedará activa hasta que alguien la lea. Así se evita que se pierdan pequeños picos de detección.

5.3 Programación de una rutina del robot

El paradigma de programación para realizar un programa es el siguiente. Se deberá programar una máquina de estados que se ejecute cada periodo de tiempo que se elija. Normalmente son unos 100ms, con ese tiempo se obtiene un tiempo de respuesta más que suficiente.

Se pueden crear las tareas que se necesiten. Por ejemplo si se quiere controlar dos partes independientes del robot, muchas veces no tiene sentido tener una única máquina de estados.

5.4 Programación del robot en tiempo real

5.4.1 Sistemas en tiempo real

La computación en tiempo real o informática en tiempo real está relacionada con los sistemas del soporte físico y la programática que se ven limitados por problemas de tiempo. El software de tiempo real debe necesariamente tener la característica de un tiempo de respuesta crítico.

Ejemplos de sistemas de tiempo real pueden ser el software encargado de controlar un respirador artificial, ya que un retraso en su tiempo de respuesta no es aceptable. Algunos tipos de programas como los empleados para jugar al ajedrez sólo disponen del tiempo necesario para poder efectuar la siguiente jugada.

Un sistema operativo en tiempo real debe cumplir ciertas características para asegurar el cumplimiento estricto de los tiempos en todo momento.

- Estos sistemas operativos suelen ser de
- De pequeño tamaño y consumen poca memoria
- Multi-arquitectura, pueden ser utilizados en distintos dispositivos.
- Tiempos de respuestas predecibles para cualquier evento.

Una de las características principales es que tienen un tiempo de respuesta igual para cualquier situación, esto no tiene que significar un tiempo bajo, simplemente un tiempo predecible. Si por ejemplo tanto si existen 10 como si existen 800 procesos, el sistema operativo debería tardar lo mismo en realizar tareas como la programación de las tareas, o la asignación de memoria. Siempre debe ser un tiempo O(1). En caso contrario existirían situaciones en las que, desde el punto de vista del diseño, no se puede predecir si se van a cumplir los tiempos estipulados.

Algunos ejemplos de sistemas operativos en tiempo real son:

- MaRTE OS
- QNX
- LynxOS
- eCos (Linux)
- VxWorks
- Windows CE
- RTAI

También existen parches para el sistema operativo Linux que intentan hacer de él un sistema operativo en tiempo real aplicándole distintos parches. No está integrado completamente, pero se planea ir incluyéndolo de manera paulatina dentro del kernel oficial de Linux. De hecho existen muchas partes del proyecto original de tiempo real, ya incluidos de manera oficial. Usando dichos parches se puede conseguir un sistema operativo en tiempo real, que aparte de ser gratuito, es Software Libre. (5).

5.4.2 Programación en tiempo real en Linux

Para programar el robot dentro de la placa principal, con el sistema operativo Linux se hará uso de distintas técnicas para alcanzar un programa que se asemeje en lo máximo posible al tiempo real.

Aunque es cierto que el robot que se pretende programar no es estrictamente necesario que sea en tiempo real, puesto que un fallo dentro del cumplimiento de los tiempos no significa un fallo del sistema. Pero sí es interesante que el tiempo de respuesta de las acciones del robot sean lo más bajos posibles, y esto se consigue mediante la programación orientada a tiempo-real.

Por ejemplo habrá tareas que comprueben que si unos sensores están activos, el robot deberá pararse

Hay que tener en cuenta varias características de la placa que pueden condicionar a conseguir un resultado que sea en tiempo real, como puede ser el sistema de almacenamiento, la memoria RAM disponible, las versiones del sistema operativo disponible.

En principio el Linux no es un sistema operativo en tiempo real, pero se asemeja bastante. Como ya se ha explicado, existen parches oficiales para hacer de Linux un sistema operativo en tiempo real. Pero debido a la complejidad de cambiar el sistema operativo en una placa embebida como es la TS-7350, se ha optado por usar el software que proporciona la empresa fabricante.

5.5 Protocolo de comunicación por el puerto serie

Uno de los problemas que existen a la hora de tener distintas placas programables en un mismo robot es la comunicación y sincronización que existe entre las mismas.

5.5.1 Protocolo de comunicaciones en el año 2009.

En el año 2009 el método que se seguía era el de enviar un comando cada cierto tiempo, cada 50ms aproximadamente. Siempre consistía en 2 bytes, que podían ser o dos velocidades, o bien posiciones de servos.

Este planteamiento presentaba varios problemas, de los que se ha aprendido bastante con el fin de mejorar el algoritmo.

Entre estos fallos, el principal era que nunca se sabía si los datos llegaban correctamente. Esto implicaba que el puerto serie se desconectaba, el programa principal no se daba cuenta.

Otro problema era que las órdenes se mezclaban, por ejemplo, si el primer byte estaba en el rango de 48 a 57 era una orden de servos, si no era de motores. Lo cuál lo hacía muy complejo, y difícil de depurar.

5.5.2 Diseño del protocolo

Según la experiencia del año anterior, se procedió a diseñar un protocolo mucho más sencillo y entendible. Que a ser posible fuese extensible a nuevas instrucciones, que a ser posible utilice bytes que estén dentro del rango ASCII de caracteres imprimibles, haciendo así más fáciles las pruebas, y más rápida la depuración.

También se ha intentado solucionar el problema de no saber cuándo se conecta usando algún sistema de confirmación de mensajes.

Se sigue usando el mismo paradigma de envío de mensajes, pero esta vez, los mensajes son de tamaño variable.

Los mensajes se componen de una cabecera de 1 byte, que identificará inequívocamente el tipo de mensaje. Después de esta cabecera podrán venir tantos bytes como sean necesarios, dependiendo del tipo de mensaje, pudiendo ser incluso 0 bytes.

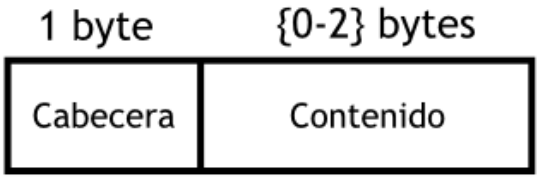

Ilustración 5-2: Esquema general de un mensaje por el puerto serie.

Todos los códigos que se han utilizado, se ha procurado que sean caracteres ASCII imprimibles.

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?
ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

Ilustración 5-3: Tabla ASCII.

Los tipos de mensaje existentes son:

Reset (Código 114. Letra r)

Cada vez que se envíe este mensaje. La placa receptora del mensaje deberá parar los motores, y dejar todos los servos en una posición neutra. Debe ser el mismo efecto que si apagásemos el dispositivo y lo volviésemos a encender.

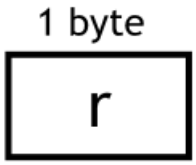

Ilustración 5-4: Esquema del mensaje Reset.

Motores. Código 109. Letra 'm'

Sirve para mandar señales de movimiento a los motores.

El contenido del mensaje serán 2 bytes, donde el primero indica la velocidad del motor izquierdo, y el segundo el del derecho.

Para controlar las velocidades negativas y positivas, se utiliza el siguiente mecanismo. Si el valor está entre 'a' y 'z' el valor es negativo. Siendo 'a' velocidad 0, y 'z' la velocidad máxima. Análogamente se utilizan las letras mayúsculas.

Por ejemplo se puede enviar el mensaje "mZZ" que indica que nos movamos hacia adelante a máxima velocidad. El mensaje "maa" o "mAA" indica que nos paremos, las dos ruedas. El mensaje "mzZ" significa girar el robot sobre su propio eje hacia la derecha (rueda izquierda hacia atrás, rueda derecha hacia adelante).

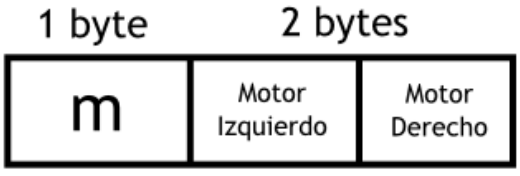

Ilustración 5-5: Esquema del mensaje Motores.

Servos, Código 115. Letra 's'.

Se utiliza para dar un ángulo a los servos. Existen 8 servos que pueden ser controlados. El primer byte será el número de servo. Para representarlos se usará el carácter ASCII '0'. y el último servo con el carácter '7'. Es importante no confundir estos valores con 0 y 7. En este caso los valores numéricos según la tabla ASCII serían desde el 48 al 55.

El segundo valor es el ángulo del servo. Este puede ser cualquier valor numérico, desde 1 hasta 256. El cero se reserva para el valor neutro. Si se indica a un servo que debe estar en la posición cero, significa que no debe hacer ninguna fuerza, quedando de esta manera suelto.

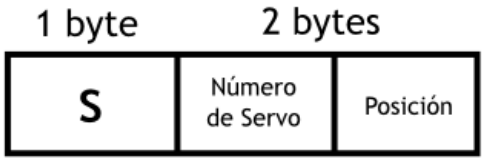

Ilustración 5-6: Esquema del mensaje de Servos.

Aserciones

Para intentar combatir dos posibles problemas. El primero es que no se sabe si un mensaje llega correctamente, y el segundo es que no se sabe si el dispositivo recptor está saturado. Por lo tanto se ha implantado un sistema de parada y espera que permita convatir estos dos problemas.

El receptor deberá mandar un mensaje de aserción indicando que ha recibido el último mensaje correctamente y que está listo para recibir el siguiente.

El mensaje de aserción estará formado únicamente por un byte que contendrá el carácter ASCII '#'.

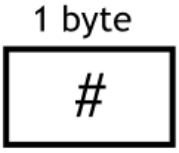

Ilustración 5-7: Esquema de una aserción.

A parte de esto hay que tener en cuenta que el propio dispositivo receptor, a cada byte que le enviamos, hace eco del mismo.

Por eso otro mecanismo de comprobación es, aparte de leer el byte de aserción, es leer todos los caracteres eco, y comprobar que corresponden con los que se han recibido.

Con el fin de no saturar al dispositivo en ningún momento, nunca se enviarán más de dos bytes consecutivos. El modo de proceder será, cada vez que se mande un byte, esperar al byte eco, y cuando se reciba enviar el siguiente, así hasta haber mandado todo el mensaje. Una vez mandado un mensaje se esperará a recibir el mensaje de aserción.

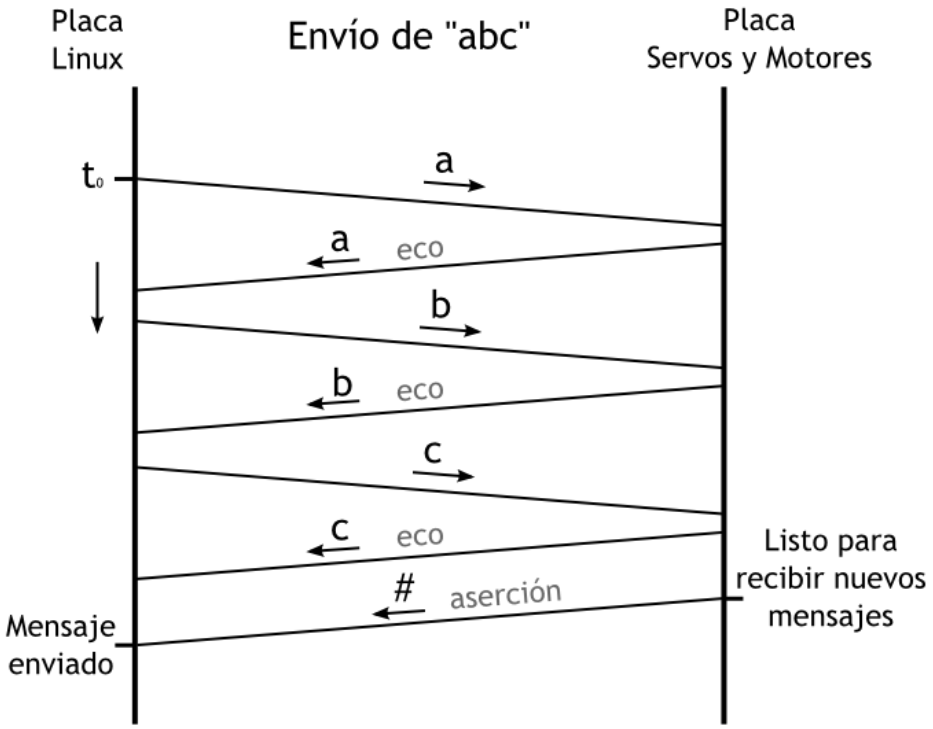

Ilustración 5-8: Ejemplo de envío de un mensaje con aserción.

5.6 Programación de una señal PWM

Una señal PWM (Pulse-width modulation), algo así como modulación por ancho de pulso. Es una manera eficiente de proveer de tensiones medias que se encuentren entre el máximo y el mínimo. (6).

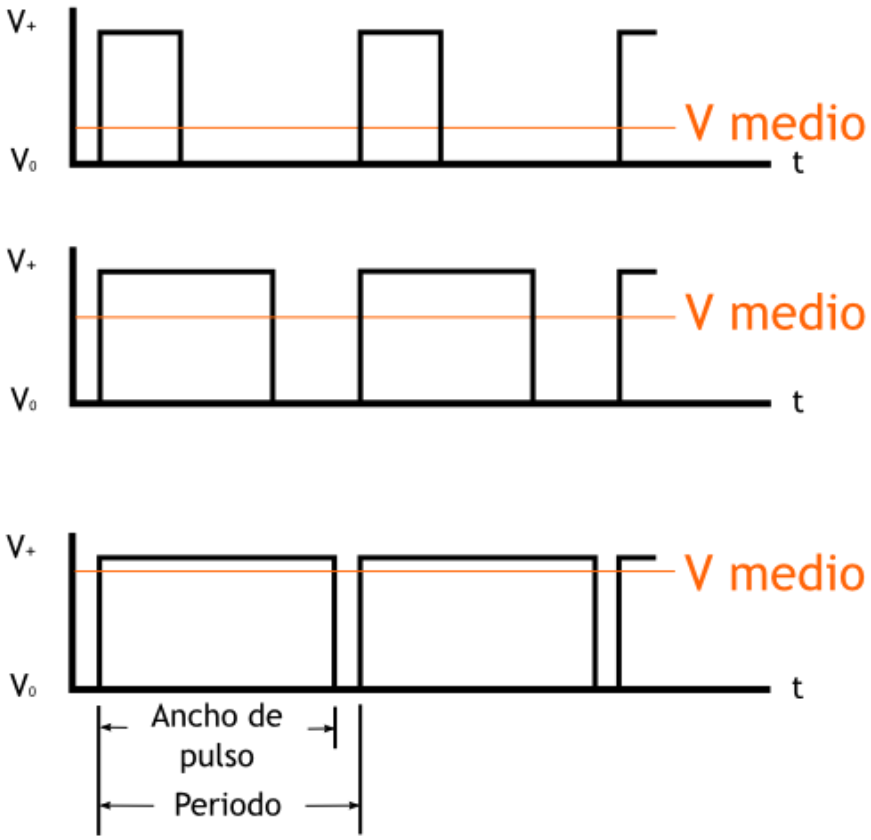

Ilustración 5-9: Distintos pulsos PWM.

Es una técnica medianamente reciente, llevada a la práctica por elementos electrónicos.

En el pasado, cuando se necesitaba regular la potencia que recibía un dispositivo, se utilizaba un potenciómetro en serie con el dispositivo a regular, que no es más que una resistencia variable. Mientras mayor fuese la resistencia, menor potencia recibía el dispositivo. Este mecanismo desperdiciaba mucha energía como resultado de estar utilizando una resistencia.

Otros dispositivos existentes para regular el voltaje, son los autotransformadores, que permitían transformar el voltaje de manera eficiente, pero su costo es demasiado elevado.

Básicamente, el esquema de un circuito PWM, cambiar rápidamente entre el voltaje máximo y el mínimo muchas veces por segundo. La frecuencia con la que se puede realizar esta operación puede ser desde 120Hz hasta miles de kHz. El ritmo de cambio de voltaje es suficientemente rápido para que el dispositivo no se vea afectado por aplicarle un voltaje demasiado alto en determinados momentos.

Una señal PWM es al fin y al cabo una señal rectangular, cuyo ancho está regulado, dando como resultado un voltaje medio, que será proporcional al ancho de la señal.

Con una señal PWM podemos controlar dispositivos, que necesitan un voltaje variable para modificar sus características. Por ejemplo los servomotores, según reciban un voltaje y otro, se posicionarán en un ángulo.

Igualmente los motores, si los alimentamos al máximo voltaje permitido, siempre obtendremos de los motores su máxima potencia y velocidad. Esto muchas veces no es lo deseable, y existen ciertos momentos en los que se necesita regular la velocidad, y para esto se necesita regular el voltaje de entrada de los motores.

Por todos estos motivos, el uso de señales PWM para regular el voltaje de entrada se hace indispensable, al ser la manera más eficiente de conseguirlo.

5.6.1 Generación de una señal PWM

Pero para generar una señal PWM correctamente se necesitan varios requisitos. Normalmente se puede diseñar un circuito que genere una señal PWM a partir de ciertos parámetros, pero también es posible generar una señal PWM usando un micro-controlador, mediante programación.

Los requisitos de un micro-controlador para generar una señal PWM es que se pueda programar, que tenga salidas digitales, y que pueda hacer uso de "timers".

La manera de generar una PWM desde un micro-controlador es haciendo uso de timers. Para esto debemos tener claros dos conceptos. El primero es el ciclo de la señal PWM, esto es el periodo de la señal. El segundo concepto es el tiempo del timer que se utiliza para controlar la señal. Este tiempo depende de dos cosas, del periodo de la señal PWM y de la resolución que queramos dar a la señal.

Imaginemos que queremos crear una señal PWM de 100Hz, con una resolución de 10. Esto significa que dentro de la señal, podremos asignarle valores de 1/10, 2/10, ó 10/10, pero nunca de valores intermedios. En este caso la frecuencia del timer debe ser de 1kHz (100Hz * 10).

La manera de generar la señal será, en cada paso del timer, decidir si se debe generar un valor '1', que significa voltaje máximo, o un '0', que significaría voltaje mínimo. Si por ejemplo queremos generar una señal que ocupe 3/10 del ancho, en los 3 primeros pasos del timer generaremos una salida '1', y en los restantes un '0'. Cada vez que se cumplan los 10 pasos del timer volveremos a empezar el ciclo de de la señal PWM.

Cuando se desea cambiar el ancho de una señal PWM no puede hacerse en cualquier momento, debe ser al finalizar el periodo de la señal. Si se cambiase en medio de la generación podría darse el caso de generar pulsos no deseados o tamaños de anchos intermedios.

5.6.2 Control de motores mediante PWM

Otro punto a tener en cuenta a la hora de generar una señal PWM es que cada dispositivo es distinto. Por ejemplo un motor se puede controlar desde el voltaje cero, hasta el máximo, pero si se le aplica un voltaje demasiado pequeño, probablemente no se moverán, o bien porque no tengan suficiente fuerza para vencer la resistencia interna del motor, o porque no tenga suficiente fuerza para mover la carga que estén soportando.

5.6.3 Control de servomotores Futaba 3003

Por otro lado los servomotores tienen requieren que las señales tengan un rango bien definido. Normalmente los servos tienen un ángulo de movimiento de 180º. Y según se aplique un voltaje determinado se posicionará en un ángulo. Pero estos márgenes suelen estar en rangos entre un 1% y un 10%.

Para los servos Futaba 3003. El rango es desde 0.3ms hasta 2.3ms con un periodo de 20ms. Este es el algoritmo que tiene aplicado en programación la placa de servos y motores. (7).

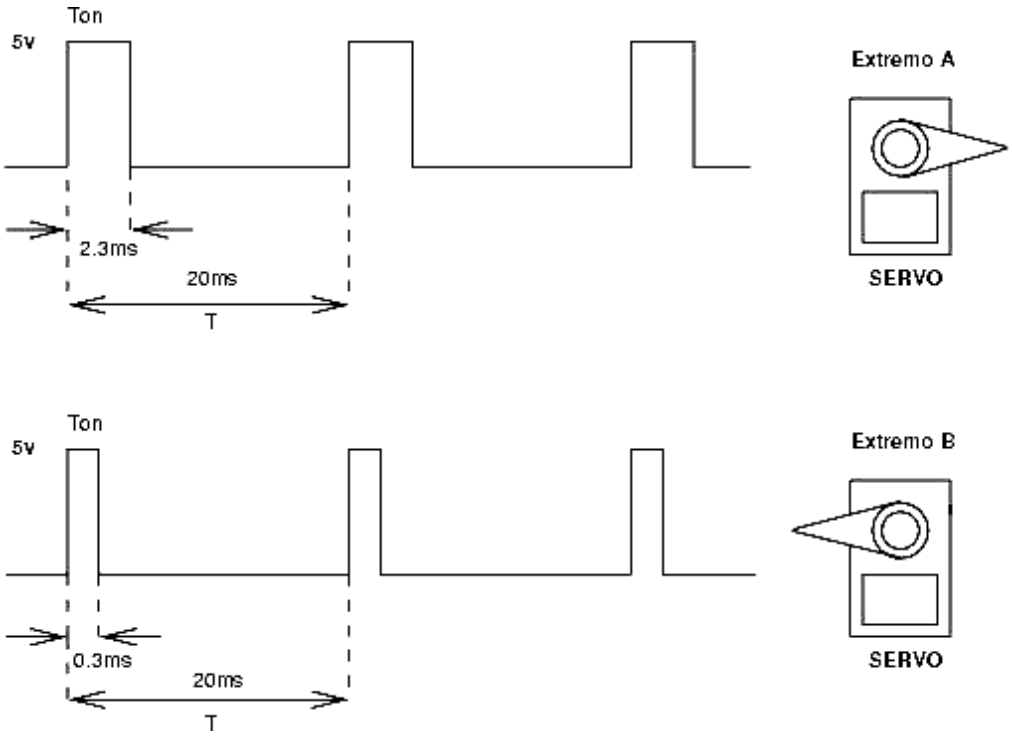

Ilustración 5-10: Señales PWM para control de un servo Futaba 3003.

5.7 Odometría

La odometría es el estudio de la estimación de la posición de vehículos con ruedas durante la navegación. Para realizar esta estimación se usa información sobre la rotación de las ruedas para estimar cambios en la posición a lo largo del tiempo. Este término también se usa a veces para referirse a la distancia que ha recorrido uno de estos vehículos.

La palabra "odometría" se compone por las palabras griegas hodos ("viajar", "trayecto") y metron ("medida"). (8)

5.7.1 Odometría en la robótica

Los robots móviles usan la odometría para estimar su posición relativa a su localización inicial, pero nunca para determinar, siempre es una estimación. Es bien sabido que la odometría proporciona una buena precisión a corto plazo, es barata de implantar, y permite tasas de muestreo muy altas. Sin embargo la idea fundamental de la odometría es la integración de información incremental del movimiento a lo largo del tiempo, lo cual conlleva una inevitable acumulación de errores. En concreto, la acumulación de errores de orientación, causa grandes errores en la estimación de la posición, los cuales van aumentando proporcionalmente con la distancia recorrida por el robot.

La odometría se basa en ecuaciones simples que se pueden implementar fácilmente y que utilizan datos de encoders situados en las ruedas del robot. Sin embargo, la odometría también está basada en la suposición de que las revoluciones de las ruedas pueden ser traducidas en un desplazamiento lineal relativo al suelo. Esta suposición no tiene una validez absoluta. Un ejemplo extremo es cuando las ruedas patinan: si por ejemplo, un rueda patina sobre una mancha de aceite y la otra no, entonces el encoder asociado registrará revoluciones en la rueda, a pesar de que éstas no correspondan a un desplazamiento lineal de la rueda. Además de este ejemplo hay muchas otras razones más sutiles por las cuales se pueden producir imprecisiones en la traducción de las lecturas del encoder de la rueda a un desplazamiento lineal. Todos estos errores se pueden agrupar en dos categorías: errores sistemáticos, y errores no sistemáticos.

Entre los errores sistemáticos destacan

- Los diámetros de las ruedas no son iguales.
- La media de los diámetros de las ruedas difieren del diámetro de fábrica de las ruedas.
- Mal alineamiento de las ruedas.
- Resolución discreta (no continua) del encoder.
- La tasa de muestreo del encoder es discreta.

Entre los errores no sistemáticos se encuentran:

- Desplazamiento en suelos desnivelados.
- Desplazamiento sobre objetos inesperados que se encuentren en el suelo.
- Patinaje de las ruedas debido a:
    - Suelos resbaladizos.
    - Sobre-aceleración.
    - Derrapes (debidos a una rotación excesivamente rápida).
    - Fuerzas externas (interacción con cuerpos externos).
    - No hay ningún punto de contacto con el suelo.

Una clara distinción entre errores sistemáticos y no sistemáticos es de gran importancia a la hora de reducir los errores en la odometría. Por ejemplo, los errores sistemáticos son específicamente graves, porque se acumulan constantemente. En muchas superficies no rugosas de entornos interiores, los errores sistemáticos contribuyen muchos más a los errores en la odometría que los errores no sistemáticos. Sin embargo, en superficies que agarran bien con irregularidades significativas, son los errores no sistemáticos los que predominan.

El problema de los errores no sistemáticos es que pueden aparecer inesperadamente, por ejemplo cuando el robot pasa por encima de un objeto que se encuentra en el suelo, y pueden causar errores muy grandes en la estimación de la posición.

Estas elipses crecen a medida que la distancia recorrida aumenta, a no ser que un sistema de estimación de la posición absoluto reduzca el crecimiento de la incertidumbre y por lo tanto ponga a cero el tamaño de la elipse de error. Estas técnicas de estimación del error se basan en estimación de parámetros derivados de los errores sistemáticos, puesto que la magnitud de los errores no sistemáticos es siempre impredecible.

5.7.2 Algoritmo de odometría

Para calcular el cambio en la posición y orientación del robot a través de un período de tiempo determinado, que se limitará sólo a la distancia lineal DR y la rueda DL cada uno ha viajado (calculada a partir del número de garrapatas de los codificadores y el diámetro de las ruedas; a calcular hacia fuera!) y conéctelo a las siguientes ecuaciones.

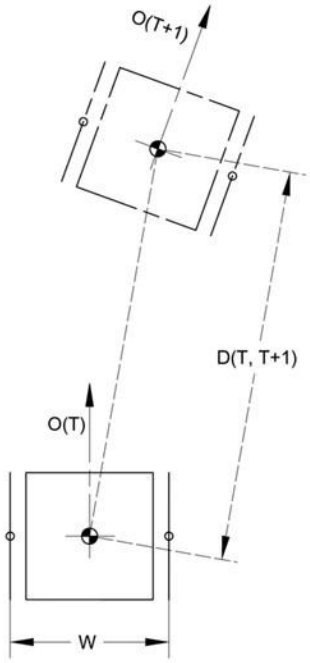

Ilustración 5-11: Esquema de un movimiento para el cálculo de la odometría.

La nueva orientación (que está en radianes) se calcula por:

$$\mathrm{O_{T+1}} = O_T + \frac{D_R - D_L}{W}$$

La distancia recorrida durante este período es:

$$\mathrm{D_{T,T+1}} = \frac{D_R - D_L}{2}$$

En el caso de la distancia recorrida, estamos calculando una aproximación lineal de una trayectoria curva. Por lo tanto, cuanto mayor sea el intervalo entre los cálculos peor va a ser esta aproximación.

Si está trabajando con un mapa (o están tratando de construir a medida que vaya de viaje), las nuevas coordenadas cartesianas del robot se calcula como:

$$\mathrm{X_{T+1}} = Y_T + D_{T,T+1} \cdot \cos(O_{T+1})$$

$$\mathrm{Y_{T+1}} = Y_T + D_{T,T+1} \cdot \sin(O_{T+1})$$

El timer interno (de tiempo T de un tiempo T, T +2, y en adelante hasta el infinito) que se usa para la odometría también se puede utilizar para recoger otras impresiones de los sentidos. Estos se pueden correlacionar con la posición del robot en el espacio y el tiempo, para construir un mapa del espacio y los acontecimientos que ocurren en él. (9)

5.7.3 Odometría como método de posicionamiento

Para poder encajar todo esto, es necesario ubicarlo en algún punto del código, y es crítico saber en qué punto de la arquitectura software debe encajar.

Dentro de la placa Linux es donde se debe llevar este control, puesto que es donde se almacena toda la lógica de control.

Dado que los cálculos se deben hacer cada vez que se leen los encoders, y los encoders son leídos desde la tarea de servos y motores. Será esta tarea la encargada de actualizar la posición del robot según las fórmulas de Odometría.

6 Conclusiones y mejoras

6.1 Conclusiones del proyecto

Para llevar a cabo este proyecto ha sido necesario 3 años de experiencia en Eurobot. El primer año como programador, aprendiendo cómo funciona la robótica. El segundo año sentando las bases de lo que es la arquitectura software. Y el tercero mejorando estas bases.

Después de este periodo es cuando se presenta este proyecto. La finalidad desde un principio, era conseguir desarrollar un software que permita aprovechar al máximo el potencial de un robot.

Se ha desarrollado una base de software lo más versátil y extensible posible, siempre dentro del paradigma de programación de un robot. Este software ha demostrado ser válido para distintos problemas, puesto que ha sido usado en dos concursos de Eurobot distintos, en el año 2009, y en el 2010.

La base de conocimiento adquirida a lo largo de este proyecto se ha centrado sobre todo en sistemas embebidos, como es el caso de un robot. La programación de micro-controladores, o de ordenadores en una única placa. Este tipo de ordenadores normalmente no se estudian a lo largo de la carrera, y probablemente se tengan pocas oportunidades en la vida como esta de utilizar este tipo de sistemas, lo que también añade más valor al conocimiento adquirido.

Los conocimientos adquiridos a lo largo de este proyecto, no se centran únicamente en la parte de programación e informática. También en el campo de la electrónica, muchas veces es fundamental entender conceptos relacionados con la electrónica, para poder crear un software que sirva a las necesidades del robot. Y debido a que en un robot se usa una programación a más bajo nivel de lo que puede estar acostumbrado un informático, se hace más necesaria esta sinergia.

Es necesario destacar también, los conocimientos mecánicos adquiridos. Todo el conocimiento de cómo se puede construir un robot desde cero. Una de las ventajas de haber desarrollado este proyecto, es que la mayoría de la parte mecánica era muy manual, lo que daba pie a la imaginación de cómo se debe hacer un robot, en lugar de encargar las piezas a una empresa, y que estos te diesen todo hecho, y no haber aprendido nada al final.

Este documento es un intento de que la parte de la informática de los proyectos Eurobot de la Universidad Carlos III de Madrid sea mejorado notablemente, y que siguientes concursantes que se decidan a empezar este mismo proyecto, tengan una base desde la que partir, que esté bien documentada, y que facilite la fase de programación, sobre todo para gente que no sea de una rama de ingeniería relacionada con la informática.

6.2 Mejoras realizadas a lo largo del proyecto

El mayor valor que se le puede dar a este proyecto es haber sido de los primeros en centrarse únicamente en la parte informática de Eurobot, que puede ser igual de importante que la parte electrónica, o la parte mecánica y de construcción.

El principal fallo que había existido hasta ahora en los proyectos de Eurobot, era la no existencia de integrantes con conocimientos de informática. Esto a la larga es un problema, porque un robot muy bien construido, con una electrónica perfecta, pero que esté mal programado, es como si no estuviese construido.

Tal como se encontraba la lógica de control del robot en el año, eran dos placas con un micro-controlador cada una, del mismo tipo y diseño, que se comunicaban entre ellas para dar órdenes a los motores y a los servos. Esta forma de partir la lógica de control en dos partes, era confusa y llevaba a error en muchos momentos.

La idea de dotar al robot con una placa más avanzada, que contuviese un sistema operativo, ha permitido, y permitirá tener una base de hardware que sea útil para cualquier año. La principal ventaja de esta placa es poder comunicarse con multitud de dispositivos de cualquier tipo. Así por ejemplo, si un año es necesario usar una cámara, no hay problema, o si se necesita manejar más de 8 servos, tampoco debería ser un problema.

Todo este software no sólo queda bien en el papel, también ha demostrado ayudar a crear programas más complejos, y por tanto, tener estrategias más logradas.

Por ejemplo se pueden controlar dos partes independientes del robot, en paralelo, cada una con su máquina de estados, sin ningún tipo de problema.

6.3 Mejoras futuras

En siguientes proyectos que se centren en desarrollar un robot para el concurso Eurobot pude ser interesante desarrollar una serie de nuevas ideas, o bien mejorar en otras ya existentes.

El primer ejemplo es la placa controladora de motores y servos. El hecho de que exista un puerto de comunicaciones serie tiene varias desventajas. La primera es que tiene una tasa de error demasiado alto. El segundo problema es el tiempo de retardo que existe desde que se informa a los motores de una acción hasta que sucede realmente, o el tiempo que se tarda en leer los encoders.

Es por esto que puede ser un tema interesante, indagar en la FPGA que contiene la placa Linux. Una FPGA es un circuito programable, así que en principio se podría crear un circuito que generase señales PWM, y que contase pulsos, que es lo que básicamente hace la placa de servos y motores.

La arquitectura software por supuesto es mejorable, se pueden añadir módulos para controlar nuevos dispositivos. O mejorar el código internamente. Para eso se creó, para que pueda evolucionar todo lo que se pueda.

7 Presupuesto

En este apartado se va hacer un cálculo del dinero que se ha invertido en construir el robot del año 2010.

Se incluyen los gastos originados por la compra de material, tanto mecánico, electrónico. La mano de obra no se cuenta como gasto ya que en este caso es llevada a cabo por estudiantes de la Universidad Carlos III de Madrid.

Los precios que se muestran son orientativos, y no tienen porqué ajustarse a la realidad, dependiendo del comercio en donde se compren, y la cantidad que se adquiera.

7.1 Estructura del robot

Concepto	**Precio Unidad**	**Cantidad**	**Total**
Lamina de policarbonato extruido, transparente de 5mm espesor.	85,6 €	2 m^2	171,20 €
Ángulo de aluminio de 3mm de espesor	2,1 €	5	10,50 €
Ángulo de aluminio de 1,5 mm de espesor	3 €	2	6 €
Material de Ferretería (Clavos, tornillos, arandelas, etc.)	10 €	1	10 €
Total Estructura			**197,70 €**

Tabla 7-1: Presupuesto de la estructura del robot.

7.2 Sistema de recogida

Concepto	Precio Unidad	Cantidad	Total
Servomotor Futaba S-3003	11 €	2	22 €
Plancha de aluminio de 1,5mm de espesor	20 €	1	20 €
Varilla de acero de 5mm de diámetro	4 €	1	4 €
Ruedas de 1cm de diámetro	4,6 €	4	18,40 €
Soportes de motores	8 €	2	16 €
Total de recogida			**60,40 €**

Tabla 7-2: Presupuesto del sistema de recogida del robot.

7.3 Sistema electrónico y de control

Concepto	Precio Unidad	Cantidad	Total
Placas de circuito impreso. Micro-controlador y Drivers.	150 €	2	300 €
Placas de circuito impreso. Potencia y acondicionamiento de señal múltiple	70 €	4	280 €
Placas de circuito impreso. Acondicionamiento de señal simple	30 €	9	270 €
Batería Yuasa NPH5-12. 12V y 5S	31,58 €	2	63,16 €
Placa base Linux	310 €	1	310 €
Micro-controlador 5082	18,3 €	2	36,60 €
Bobina de conexionado varios	1,32 €	3	3,96 €
Bobinas cable de alimentación 0,5mm	1,4 €	2	2,80 €
Sensor de infrarrojos GP2D12	14,56 €	15	218,40 €
Sensor de infrarrojos GP2D120	18,68 €	5	93,40 €
Sensor de final de carrera	1,3 €	2	2,60 €
Pulsador de parada de emergencia	19,48 €	1	19,48 €
Total sistema electrónico y de control.			**1600,40 €**

Tabla 7-3: Presupuesto del sistema electrónico.

7.4 Sistema locomotriz

Concepto	Precio Unidad	Cantidad	Total
Motores con reductora y encoders incorporados	178 €	2	356 €
Pack 4 ruedas de patín	25 €	1	25 €
Rueda loca trasera	4,7 €	1	4,70 €
Casquillos	30,3 €	2	60,60 €
Soportes de motores	8 €	2	16 €
Total sistema locomotriz			**462,30 €**

Tabla 7-4: Presupuesto del sistema locomotriz del robot.

7.5 Campo de pruebas

Concepto	Precio Unidad	Cantidad	Total
Tablero para el campo	550 €	1	550 €
Listones de madera	3 €	10	30 €
Material de ferretería	25 €	1	25 €
Pintura. Verde, Azul, Amarilla y Negra	25 €	4	100 €
Placas de madera de conglomerado para elementos anejos	16 €	$4m^2$	64 €
Total sistema locomotor			**769 €**

Tabla 7-5: Presupuesto del campo de pruebas para el robot

7.6 Costes de personal

Se ha incluido el precio de personal de trabajo, de cuatro ingenieros técnicos industriales, y de dos ingenieros informáticos. Se calcula un tiempo estimado de ejecución de 8 meses, con una media de 6 horas diarias.

También se incluye el trabajo de personal becario, con una media de 2 horas diarias durante los 8 meses.

Cargo	**Número de personas**	**Número de horas por persona**	**Precio por hora**	**Total**
Ingeniero técnico industrial	4	1056	8 €	33792 €
Ingeniero informático	2	1056	9 €	16896 €
Personal ayudante	2	352	4 €	2816 €
Total sistema locomotor				**53504 €**

Tabla 7-6: Presupuesto de personal

7.7 Presupuesto total Eurobot 2010

Concepto	Total
Estructura del robot	197,70 €
Sistema de recogida	60,40 €
Sistema electrónico y de control	1600,40 €
Sistema locomotriz	462,30 €
Campo de pruebas	769 €
Costes de personal	53504 €
Total presupuesto Eurobot 2010	**56593,80 €**

Tabla 7-7: Presupuesto total Eurobot 2010

El presupuesto total para la realización del proyecto Eurobot 2010 es de **cincuenta y seis mil quinientos noventa y tres euros con ochenta céntimos** (56593,80 €).

Anexos

1 Creación de un programa para Linux usando el entorno Eclipse

1.1 Programación para Linux

Debido a que se trata de una arquitectura distinta a un equipo corriente, es necesario el uso de compiladores cruzados, que permiten compilar una aplicación en una arquitectura como puede ser Intel, para otra arquitectura distinta, en este caso ARM. Esto es necesario en el caso que se desee usar un lenguaje compilado como puede ser C o C++.

También se puede hacer uso de otro tipo de lenguajes interpretados, como pueden ser Java, Perl o Python. Al tratarse de un sistema operativo Linux Debian, la instalación de los intérpretes de estos lenguajes es sencilla, y el proceso de compilado, o bien no es necesario, como en Perl y Python, o bien se hace a un lenguaje intermedio como en Java.

Otra opción consiste en compilar las aplicaciones que se desean dentro de la propia placa, en donde se encuentran los compiladores de C y C++ ya instalados. Este procedimiento de compilación no es el más cómodo, pero permite compilar pequeñas aplicaciones rápidamente, ayudado por el servidor FTP para transmitir los ficheros. Si se hace uso de características que existen en sistemas operativos Linux de editar los ficheros alojados en un servidor FTP directamente, junto con la línea de comandos por Telnet, se hace un proceso más sencillo.

Para hacer todo este proceso más sencillo, la empresa Technologic Systems, la empresa que fabrica la placa Linux, provee un entorno de desarrollo basado en eclipse que facilita la programación, compilación, y transferencia a la placa Linux de los programas generados.

Este entorno de desarrollo se puede descargar directamente desde el servidor FTP de la empresa Technologic Systems. Está pensado para ejecutarse sobre plataformas Windows, y viene con todo lo necesario para empezar a desarrollar, como pueden ser compiladores cruzados, consola Telnet, cliente FTP.

1.2 Entorno de desarrollo Eclipse.

1.2.1 Software Eclipse

Eclipse es una comunidad de código abierto, cuyos proyectos se centran en la creación de una plataforma de desarrollo abierta formada por aplicaciones extensibles, herramientas y runtimes para crear, desplegar y gestionar software en el ciclo de vida. La Fundación Eclipse es una organización sin fines de lucro, corporación de miembros que alberga los proyectos de Eclipse y ayuda a cultivar tanto la comunidad de código abierto y un ecosistema de productos y servicios complementarios.

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. La Fundación Eclipse fue creada en enero de 2004 como una entidad independiente sin fines de lucro para actuar como administrador de la comunidad Eclipse. Hoy, la comunidad Eclipse se compone de individuos y organizaciones de una sección transversal de la industria del software. (10).

Al tratarse de marco de trabajo ampliable, la aplicación Eclipse puede ser usada, si se instalan los complementos necesarios, para otros fines distintos para los que se pensó originalmente. Es raro el lenguaje que no tenga un complemento para poder programar en Eclipse.

1.2.2 Familiarización con el entorno Eclipse para Linux

El primer paso de todos, si no se ha hecho ya es descargar el software Eclipse para el desarrollo, que la empresa Technologic Systems proporciona.[1]

Igualmente este software viene incluido al comprar la placa, si se adquiere con una tarjeta SD.

Ilustración 1-1: Icono de Eclipse.

[1] ftp://ftp.embeddedarm.com/misc/eclipse/

La primera vez que abramos el software podremos comprobar que tenemos a la izquierda los proyectos existentes, abajo a la izquierda los ficheros en el sistema remoto, que es la placa Linux, y abajo a la derecha la consola Telnet y más información. En el centro es en donde se ven los archivos fuente que se están editando.

Arriba también tenemos distintas barras de herramientas, con las configuraciones de Eclipse por defecto.

Pronto se podrá comprobar que el entorno Eclipse es muy cómodo para trabajar. Permitiéndonos compilar, copiar los programas hacia la placa, y ejecutarlos de una manera muy rápida y cómoda.

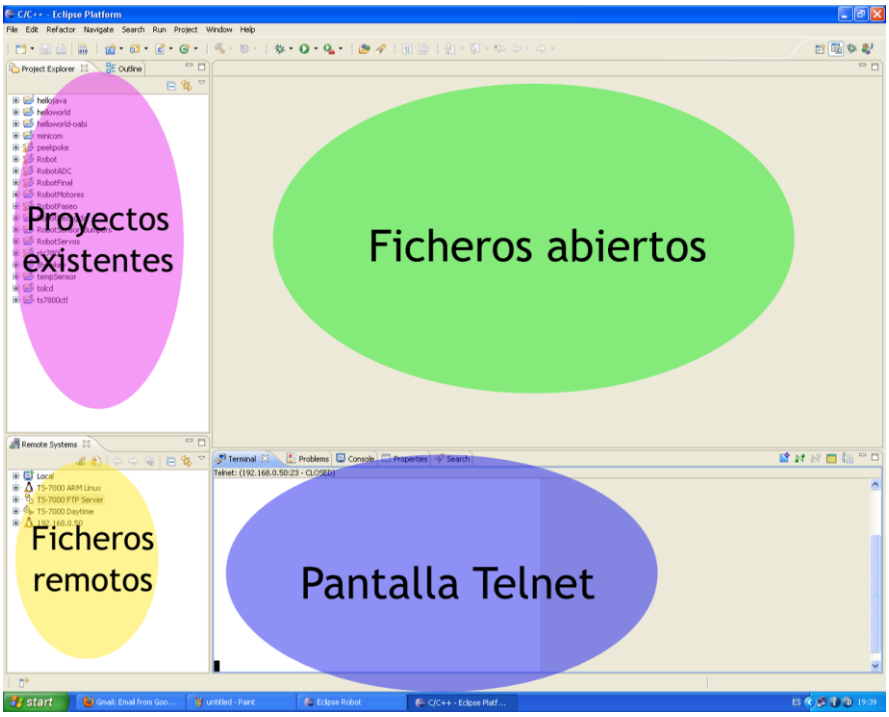

Ilustración 1-2: Distribución del entorno de desarrollo Eclipse.

1.2.3 Conectar con la placa por Telnet

Una de las características que nos ofrece este entorno es conectarnos mediante una pestaña por Telnet a la placa. Normalmente se podría usar otro software como el propio programa "telnet" de Windows, o programas como PuTTY[2].

La principal ventaja de este mecanismo es que no necesitamos configurar ninguna IP, ya que viene configurada por defecto a la IP 192.168.0.50, que es la IP que la placa Linux usa.

Ilustración 1-3: Conectando mediante Telnet.

Una vez pulsado el botón "Connect", se podrá verificar que enseguida aparece un mensaje de bienvenida. En caso de que la placa no esté conectada, o esté apagada, aparecerá un mensaje de "timeout".

1.2.4 Compilar un proyecto

Como ya se ha explicado, el entorno Eclipse nos proporciona muchas facilidades, y no iba a ser menos para la compilación.

Para realizar la compilzación de un proyecto, simplemente debermos pulsar el botón "Build" que se encuentra en la barra de herramientas.

[2] http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

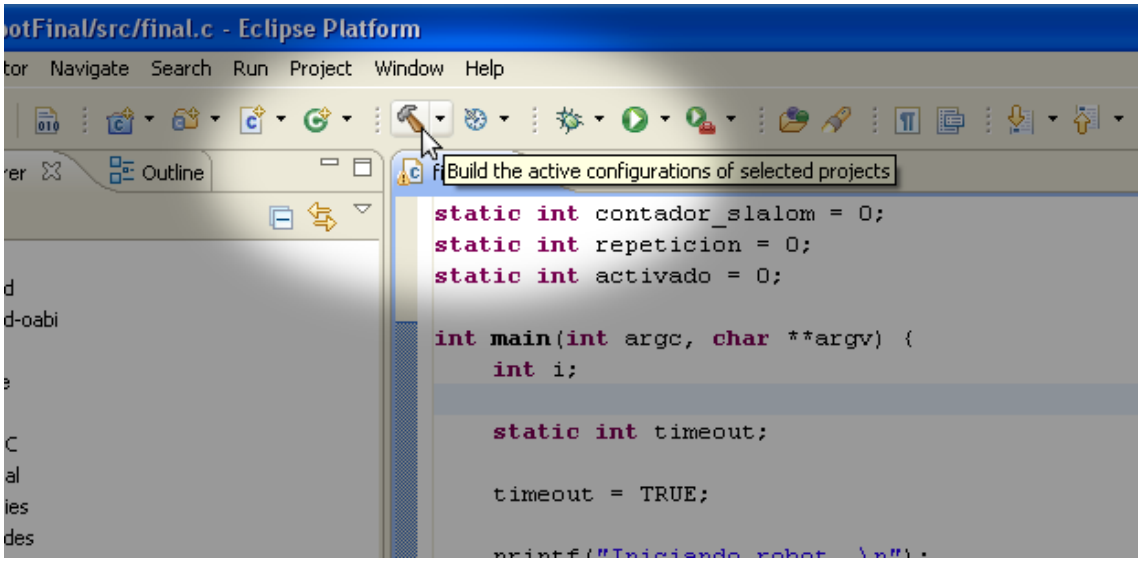

Ilustración 1-4: Botón de compilar en Eclipse.

Puede que la primera vez la compilación tarde un poco, pero no suele tardar más de 3 ó 4 segundos.

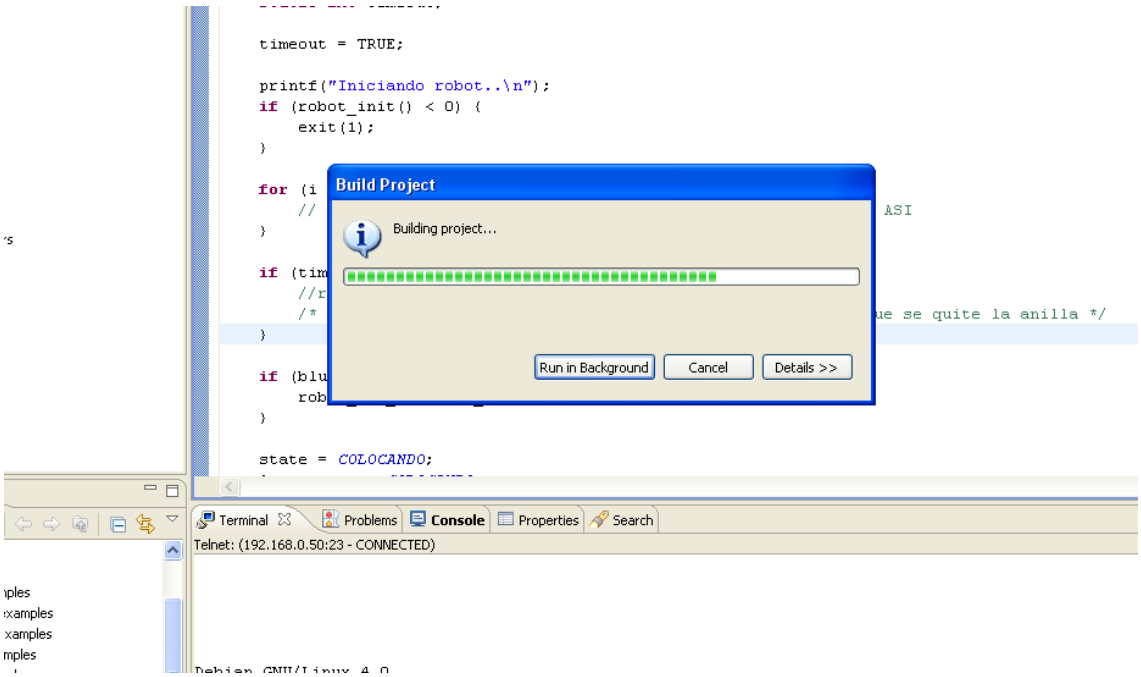

Ilustración 1-5: Compilando un proyecto en Eclipse.

Una vez que la compilación termine, para comprobar que ha sido satisfactoria debemos fijarnos en la pestaña "Console" de la parte inferior.

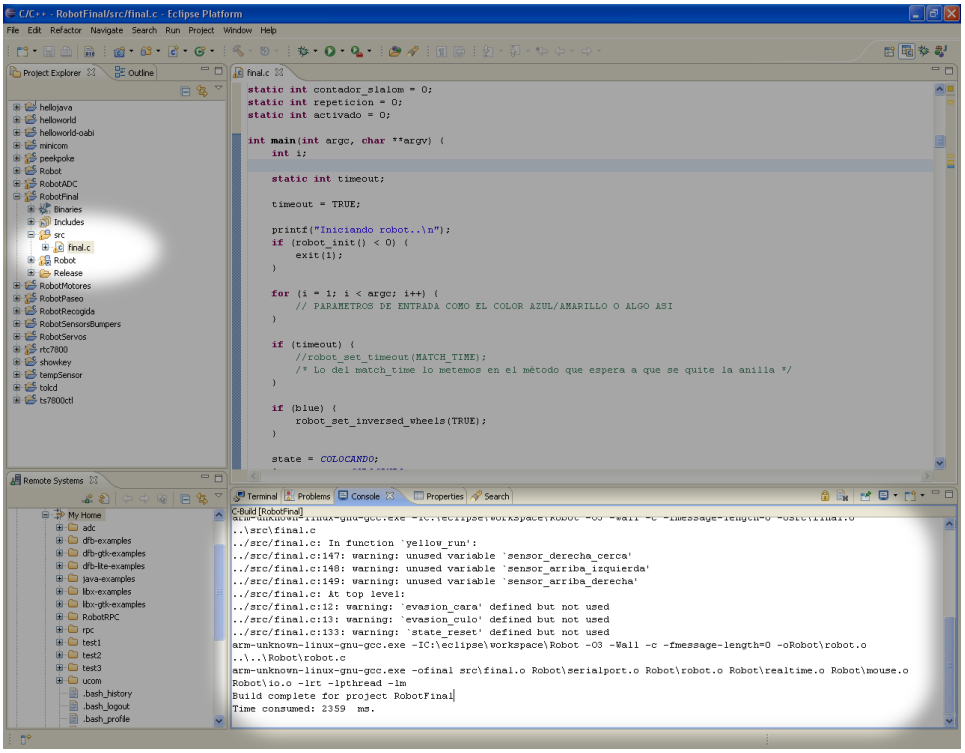

Ilustración 1-6: Compilación correcta.

En caso de que se produzca algún error, se mostrará la línea donde se ha producido.

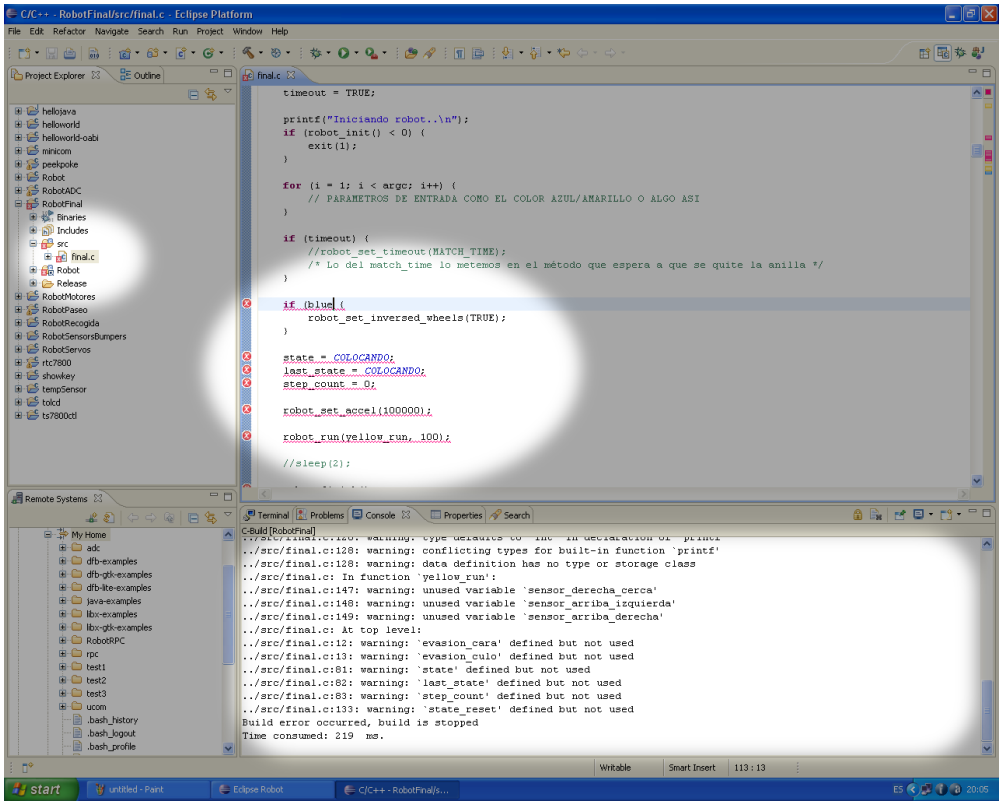

Ilustración 1-7: Compilación con errores.

2 Reinstalación de la placa Linux

Para más información sobre la placa Linux, remitirse al apartado "0

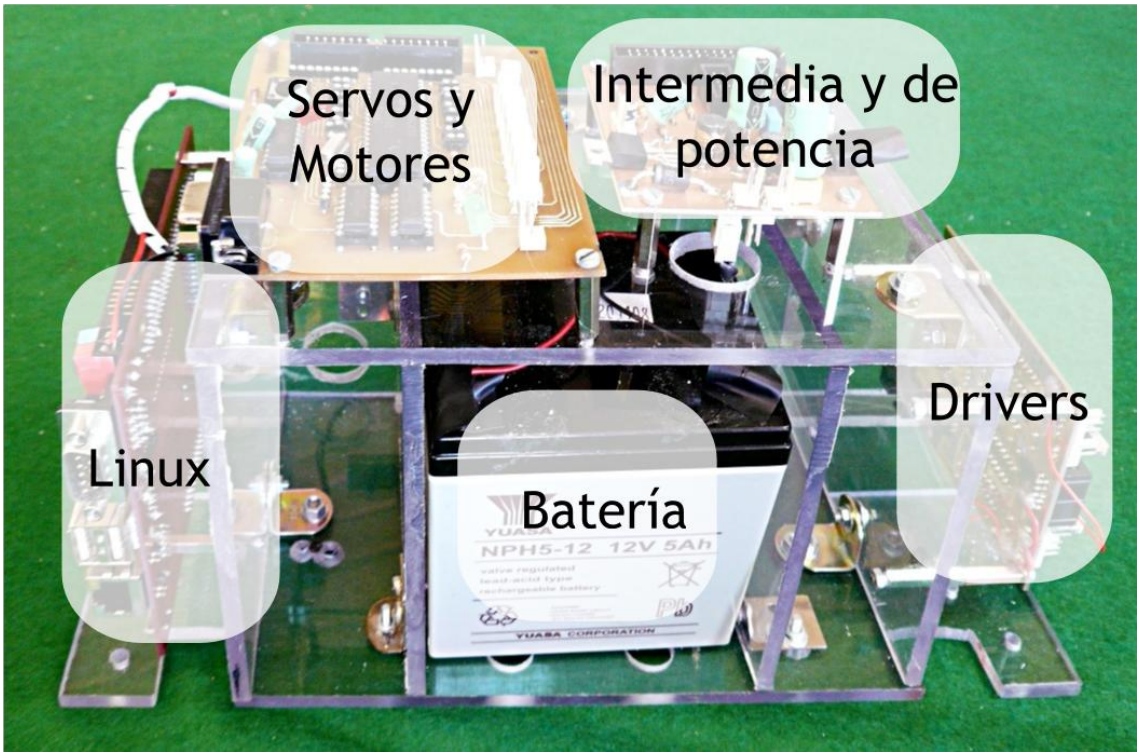

Ilustración 4-2: Esquema sobre el esqueleto del robot.

Placa principal Linux"

2.1 Instalación desde cero del software

La placa necesita de un sistema operativo para funcionar. Existen placas con una memoria PROM donde se almacena el Sistema Operativo. En el caso del modelo TS-7350 es necesario proporcionar una tarjeta SD que incluya el Sistema Operativo.

El contenido de la placa debe tener una configuración de particiones específica. La tarjeta se compone de cuatro particiones (11).

La primera partición es un sistema de ficheros FAT32 vacía en principio, aunque puede contener datos como documentación, etc. En la tarjeta SD que se compra directamente al proveedor, se incluye toda la documentación, así como una versión de eclipse especialmente pensada para programar usando esta placa.

La segunda partición contiene una imagen del kernel en bruto, quiere decir que no existe ningún sistema de ficheros en esta partición, y será la que lea la placa nada más arrancar.

La tercera partición contiene un sistema de ficheros con una Sistema Operativo básico llamado busybox. Este sistema operativo permite arrancar los servicios básicos en muy poco tiempo. Puede estar arrancado desde que se enciende en aproximadamente un segundo.

Lo primero de todo es hacer un volcado en bruto sobre la tarjeta. Esto se hace con el fichero 512mbsd-feb022010.dd en este caso. Este fichero se puede obtener directamente desde el FTP de Technologic Systems [3].

Si el fichero descargado está en formato .bz2 lo primero que debemos hacer es descomprimirlo con

```
bunzip2 512mbsd-feb022010.dd.bz2.
```

El volcado se realiza con el siguiente comando.

```
dd if=512mbsd-feb022010.dd of=/dev/sdc.
```

Este paso es muy importante y hay que tener claro que el disco duro escogido es el correcto, si este paso no se realiza la placa será incapaz de arrancar de esta tarjeta.

Con esto queda instalado todo el sistema de ficheros listo para que arranque la placa. Pero antes de nada hay que retocar ciertos aspectos que por defecto están mal configurados.

Se pueden instalar cada parte indistintamente, si se desea, por ejemplo, reinstalar únicamente el kernel. Pero lo recomendado es reinstalar todo, porque si no se pueden producir fallos inesperados.

2.2 Configuración de Busybox y Debian

Dentro del busybox, existen varios scripts de inicio, por defecto en la instalación hecha en este manual, viene configurado para usar linuxrc-fastgui.

[3] ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7350-linux/binaries/ts-images/

Lo correcto sería usar linuxrc-sdboot, que arranca el sistema operativo debian incluido en la tarjeta SD.

Para ellos hacemos que linuxrc apunte a linuxrc-sdboot.

```
ls -sf linuxrc-sdboot linuxrc
```

Se puede crear un script que sea más sencillo, únicamente para los propósitos de un robot.

Tampoco existe ningún problema en dejar "linuxrc-fastboot" como script de inicio. En este caso se ejecutará Busybox al incio, pero si se desea iniciar Debian se puede hacer de la siguiente manera.

Se debe buscar el proceso "sh –i"

```
ps ux
```

Hay que fijarse en el PID que tiene este proceso para matarlo.

```
Kill -9 <PID>
```

Donde <PID> es el PID del proceso que hemos visto antes.

Una vez se ejecute este comando, automáticamente se cerrará la conexión Telnet actual, y

Esto es útil si, por ejemplo, se quiere Busybox para ejecutar aplicaciones de manera rápida, y utilizar Debian para programar.

La versión de Debian por defecto instalada, tiene demasiados servicios que no son necesarios. Por ejemplo apache2, ssh, o portmap. Si no vamos a usar ninguno de estos servicios, lo mejor es desactivarlos o incluso desinstalarlos.

```
update-rc.d apache2 disable
```

Para desactivar apache2.

2.3 Encendido y configuración de LEDs

Cuando se enciende la placa, no se enciende por defecto ningún LED, esto no sé sabe si es un error de fábrica y que el diseño original es así. Sin embargo los LEDs funcionan correctamente, por lo que modificaremos los ficheros de inicio para conseguir esto.

Cuando se inicial busybox, el fichero que debemos tocar será linuxrc-sdboot.

Dentro de debian, el fichero que se ejecuta al iniciar es "/etc/rc.local"

Los LEDs es una buena manera de conocer el estado de la placa, por lo que es interesante establecer un lenguaje que permita saber en cualquier momento, en qué estado se encuentra la placa. Hay que tener en cuenta que con tres LEDs no se puede hacer mucho.

Un posible diagrama de estados puede ser el siguiente:

Ningún LED encendido: Algo malo pasa, porque en cualquier estado se intentará encender algún LED.

LED verde y rojo encendidos. Al iniciar el sistema operativo Busybox, se modificará el script para que encienda estos dos LEDs, indicando que la secuencia de arranque ha sido correcta.

Sólo LED verde encendido. Cuando se inicie el sistema operativo Debian, se apagará el LED rojo, indicando que se ha terminado de arrancar Debian. Este pequeño procedimiento tiene su utilidad, debido a que no existe otra manera de saber cuándo se ha terminado de iniciar Debian. Además este proceso suele tardar alrededor de un minuto.

El LED amarillo se queda libre para mostrar trazas del programa que ejecutará el robot. Por ejemplo puede indicar que el programa está ejecutándose, o que está esperando a que quitar la anilla.

3 Normas Eurobot

3.1 Reglamento común a todos los años

Sólo se permite un robot por equipo.

Los robots deben ser completamente autónomos, no se permiten comunicaciones con elementos externos, excepto con las balizas.

Juego limpio

La finalidad del concurso es jugar el máximo número de partidos, de manera amistosa. De esta manera no se permite, y estará penalizado:

- Bloquear al robot contrario el acceso a los elementos del juego, así como impedir deliberadamente su movimiento.
- Diseñar el robot con el fin de confundir al contrario usando colores designados para los elementos del juego, así como partes del tablero.
- Hacer daño intencionado al robot contrario.

Seguridad

Los robots deben evitar tener partes puntiagudas o afiladas que puedan dañar al robot contrario.

Está prohibido el uso de líquidos, productos corrosivos, materiales pirotécnicos, o seres vivos dentro del robot.

Material obligatorio

Un cordón de al menos 50cm que se debe utilizar para arrancar el robot. Este cable debe ser la manera de hacer que el robot comience el partido, y al tirar de él, debe quedar completamente desacoplado del robot.

Botón de parada de emergencia, de al menos 2 cm de diámetro. Debe ser rojo, y al pulsarlo, el robot debe pararse inmediatamente, cortando el suministro de energía a cualquier parte mecánica del robot. Debe situarse en la parte superior del robot, siendo accesible en cualquier momento por el árbitro.

Apagado automático que actúe al transcurrir **90 segundos** tras el comienzo del partido. En ese momento, el robot debe cortar el suministro de energía a todas las partes mecánicas del robot.

Sistema de evasión de obstáculos que impida que el robot colisione contra un robot contrario. No es necesario esquivarlo, simplemente detectarlo a una distancia razonable para no chocarnos con él. Siempre se toma como modelo de robot un cilindro de 30cm por 20 cm de diámetro.

Soporte para baliza

El robot debe disponer de un soporte, donde el robot contrario podrá situar una baliza si así lo desea.

No es obligatorio tenerla, se puede homologar sin ella, pero en caso de que en un partido se compita contra un equipo que desee colocar una baliza en el robot contrario, se descalificará automáticamente al robot que no tiene soporte para baliza.

Dimensiones del robot

El tamaño de la base del robot debe tener dos estados.

El primer estado será el **estado replegado**, y será en el que empezará el robot. En este estado el robot no puede superar los **120cm de diámetro**.

Una vez comienza el partido el robot podrá pasar a un **estado desplegado**, donde su diámetro máximo podrá ser de **140cm**.

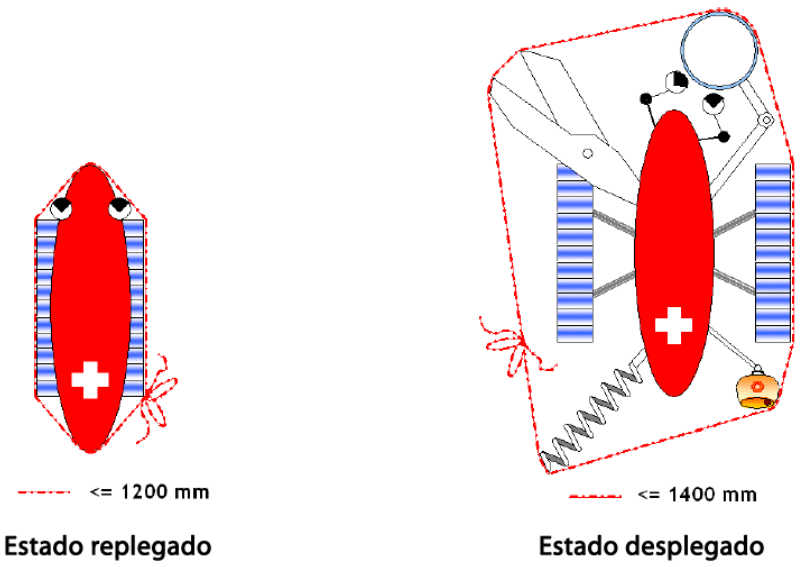

Ilustración 3-1: Dimensiones máximas de la base del robot.

En ambos casos, el diámetro se mide usando una cinta métrica, o una tela con las dimensiones indicadas.

En cuanto a la **altura**, el tope máximo permitido es de **35 cm**, pudiendo llegar a 43 cm al añadir el soporte para baliza. Si el robot adversario coloca en este soporte su baliza, la altura del robot llegará a 51 cm.

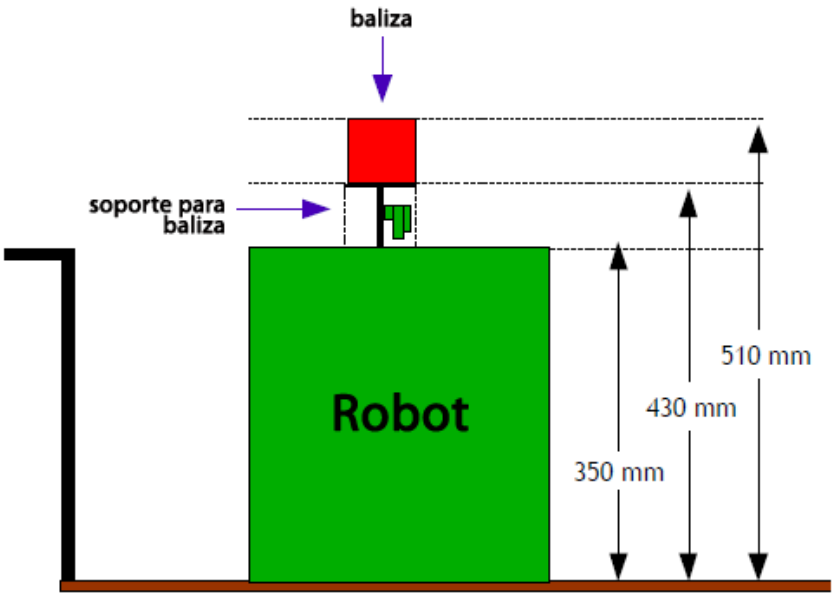

Ilustración 3-2: Altura máxima del robot.

Penalizaciones

Cualquier acción incompatible con el espíritu de juego limpio que se fomenta en el concurso, será penalizado por los árbitros. Por ejemplo, los siguientes actos serán castigados:

- Un robot choca violentamente contra otro.
- Cuando un robot sea considerado peligroso para el robot adversario, para el campo de juego o para el público.
- Si un robot impide deliberadamente al robot oponente acceder a algún elemento del campo de juego.
- Cuando un robot arroja continuamente elementos fuera del campo de juego.

Descalificaciones

Un equipo será descalificado **de ese partido**, cuando alguna de estas situaciones ocurra:

- El equipo no llega a tiempo a la sala de espera previa a un partido donde se reúne a los equipos participantes.
- El equipo no consigue estar preparado en el campo de juego en menos de 3 minutos.
- El robot no llega el soporte para balizas y el equipo adversario solicita colocarle una baliza.
- El robot no consigue abandonar completamente el área de inicio.

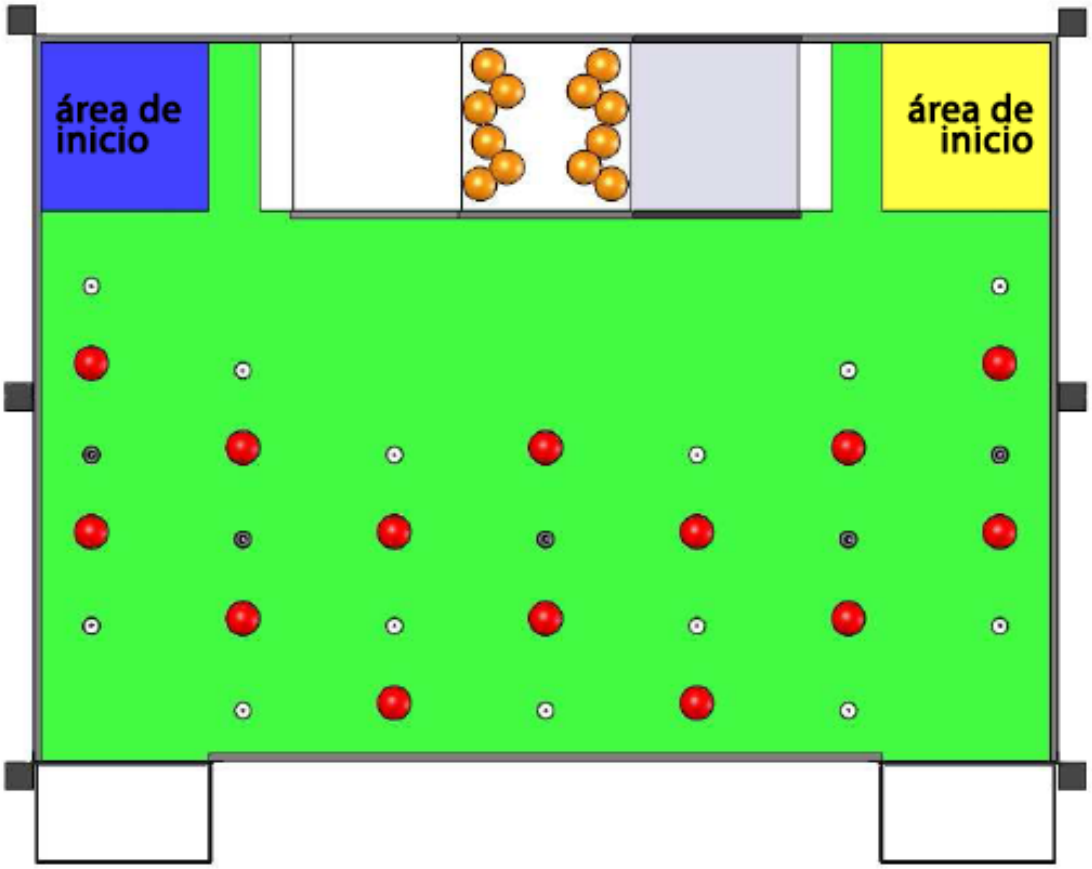

Ilustración 3-3: Area de inicio del campo de juego

Un equipo será descalificado **de la competición**, cuando:

- El robot realice reiteradamente la misma acción penalizable.
- El equipo tenga un comportamiento inaceptable.
- El robot no cumpla con las normas de seguridad establecidas.

Tamaño del campo de juego

Los encuentros se llevarán a cabo en un campo rectangular de **210 cm x 300 cm**, pudiendo ser aún mayor en algunas partes para albergar distintos elementos imprescindibles para el juego.

3.2 Reglamento específico Eurobot 2008. Mission to Mars

Aquí se detallan las características especiales del año 2008, teniendo en cuenta que las normas del anterior apartado son completamente válidas, y estas normas, únicamente se apoyan en las anteriores ya descritas.

Ilustración 3-4: Logotico de Eurobot 2008.

Campo de juego

El tablero de juego será de color gris. Se compondrá de:

- ***45 pelotas***. Las pelotas rojas son para el equipo rojo. Las pelotas azules son para el equipo azul. Las pelotas azules y rojas simbolizan las pruebas de vida. Las pelotas blancas son válidas para ambos equipos, simbolizan el hielo.

- ***4 dispensadores verticales***: 2 contienen pelotas blancas, otro contiene pelotas rojas y otro contiene pelotas azules.

- ***1 dispensador horizontal***: contienen 6 pelotas rojas y 6 pelotas azules.

- ***2 contenedores estándar***: uno para pelotas rojas y otro para pelotas azules. Se encuentran al nivel de altura del campo de juego.

- ***2 contenedores helados***: uno para pelotas rojas y otro para pelotas azules. Ubicados en los lados más cortos del campo de juego.

- Las 45 pelotas están distribuidas en el campo de la siguiente manera:

    - 13 sueltas por el tablero de juego (su distribución se realiza por sorteo): 9 blancas, 2 rojas y 2 azules

    - 20 pelotas en los dispensadores verticales

    - 12 pelotas en el dispensador horizontal

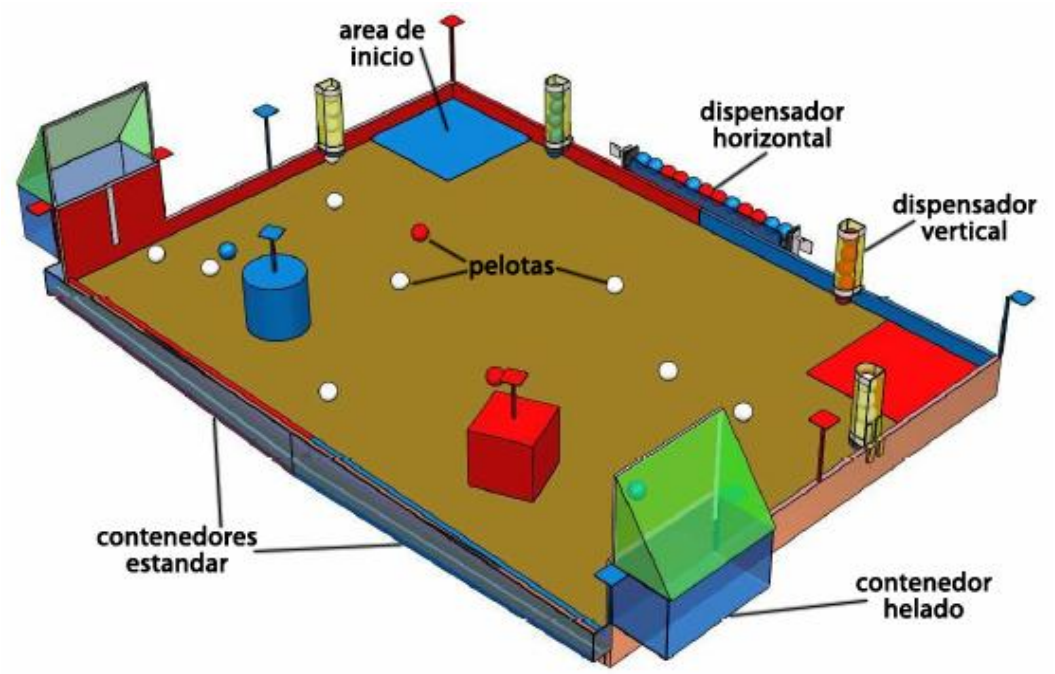

Ilustración 3-5: Campo de juego Eurobot 2008.

Puntuación

Cada equipo debe recoger las pelotas de su color (o también las blancas) que se encuentran en los dispensadores o sueltas en el tablero de juego y depositarlas en cualquiera de los contenedores (estándar o helado) ***de su color***, es decir, el equipo rojo en los contenedores rojos y el equipo azul en los azules. Al terminar los 90 segundos del partido, se hará el recuento de la siguiente forma:

- Las ***pelotas blancas*** (hielo) que se encuentren en los contenedores azules/rojos sumarán ***1 punto*** para el equipo azul/rojo

- Las ***pelotas azules*** que se encuentren en ***cualquier contenedor estándar*** (ya sea el rojo o el azul) sumarán ***2 puntos*** para el equipo azul.

- Las ***pelotas rojas*** que se encuentren en ***cualquier contenedor estándar*** (ya sea el rojo o el azul) sumarán ***2 puntos*** para el equipo rojo.

- Las ***pelotas azules*** que se encuentren en el ***contenedor helado azul*** sumarán ***2 puntos*** para el equipo azul.

- Las ***pelotas rojas*** que se encuentren en el ***contenedor helado rojo*** sumarán ***2 puntos*** para el equipo rojo.

- Cualquier pelota ubicada en un ***contenedor helado que no sea de su color***, ***restará 1 punto*** al equipo del color de la pelota.

3.3 Reglamento específico Eurobot 2009. Temples of Atlantis

Ilustración 3-6: Logotipo Eurobot 2009.

Campo de juego

El campo de juego es de color azul. Se compone de:

- ***2 zonas de construcción de nivel 1***. Ubicadas en el lado opuesto a las áreas de inicio.
- ***1 zona de construcción de nivel 2***. Situado entre las dos zonas de nivel 1.
- ***1 zona de construcción de nivel 3***. De forma circular, ubicada en el centro del tablero.

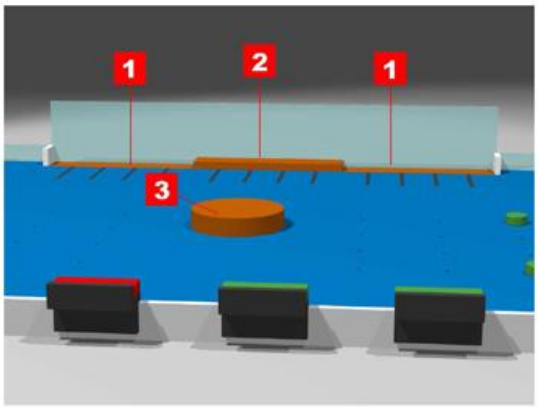

Ilustración 3-7: Distintos niveles de la zona de construcción.

- ***32 cilindros para construir columnas***: 16 de color rojo (para el equipo rojo) y 16 de color verde (equipo verde)

- ***4 dinteles***: 2 rojos (equipo rojo) y 2 verdes (equipo verde). Cada robot además podrá opcionalmente llevar pre cargado un dintel de su color.

- ***4 dispensadores verticales de cilindros***. Dos de ellos contendrán 8 cilindros de color rojo, y los otros dos 8 cilindros de color verde.

- ***4 dispensadores de dinteles***. Cada uno de ellos contendrá un dintel, dos de ellos un dintel de color rojo, los otros dos de color verde.

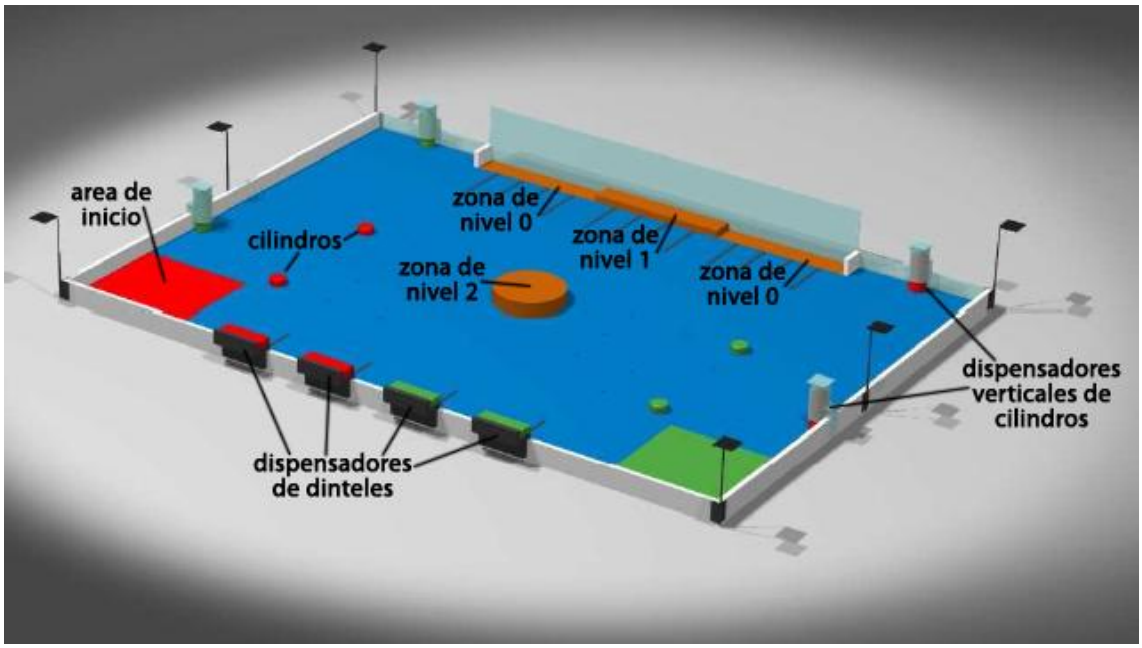

Ilustración 3-8: Campo de juego Eurobot 2009.

Puntuación

Cada equipo debe construir la mayor cantidad de templos y columnas que le sea posible, utilizando los cilindros y dinteles de su color que se encuentran las distintas partes del campo de juego (dispensadores o tablero).

Las construcciones deben llevarse a cabo en cualquiera de las zonas de construcción. La puntuación obtenida por cada columna o templo dependerá del nivel de la zona donde se haya construido; a más nivel, más puntuación.

Al terminar los 90 segundos del partido, se hará el recuento. Será considerado válido para puntuar:

- Cualquier cilindro que se encuentre completamente dentro de una de las zonas de construcción, apoyado sobre la misma.

- Aquel cilindro que se encuentre completamente dentro de una de las zonas de construcción, apoyado en otro elemento considerado válido para puntuar (otro cilindro o dintel).

- Un dintel que se encuentre completamente dentro de una de las zonas de construcción, apoyado (por uno de sus dos lados más grandes) en al menos dos elementos considerados válidos para puntuar (cilindro o dintel).

No es necesario que los elementos se apoyen sobre otros del mismo color para ser considerados válidos. El cilindro verde de la siguiente ilustración será válido.

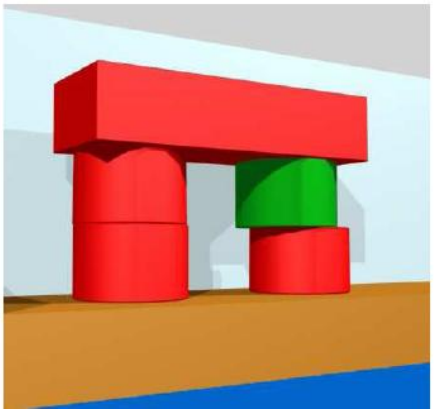

Ilustración 3-9: Situación de puntos válida

Los distintos elementos considerados válidos, puntuarán de la siguiente manera:

- Cada cilindro válido apoyado directamente sobre una zona de construcción obtendrá tantos puntos como el nivel en el que esté construido. Por ejemplo, un cilindro construido en una zona de construcción de nivel 1 obtendrá 1 punto.

- Cada cilindro válido apoyado sobre otro elemento válido obtendrá tantos puntos como el nivel al que esté construido, más el número de alturas que tenga debajo. Por ejemplo, un cilindro construido en una zona de nivel 3, apoyado sobre otro cilindro que está apoyado directamente sobre la zona de construcción obtendrá 3 + 1 = 4 puntos.

Cada dintel válido obtendrá la puntuación de forma análoga a los cilindros, multiplicando por 3 la puntuación final. Por ejemplo, un dintel construido en una zona de nivel 2, apoyado sobre dos cilindros que están apoyados directamente sobre la zona de construcción obtendrá (2 + 1) x 3 = 9 puntos.

3.4 Reglamento específico Eurobot 2010. Feed The World

Ilustración 3-10: Logotipo Eurobot 2010.

Campo de juego

El tablero de juego es de color verde. Se compone de:

2 contenedores. Uno para cada equipo. Cada equipo deberá depositar en su contenedor correspondiente los distintos alimentos que vaya recolectando.

12 naranjos. Situados en 2 grupos de 6, en la parte alta de una cuesta. Cada grupo está ubicado próximo al área de inicio de cada uno de los equipos. Cada naranjo contiene una naranja. Por lo tanto, habrá 12 naranjas en juego, representadas por pelotas de malabares de color naranja.

14 tomates. Representados por pelotas de color rojo. Distribuidos a lo largo del tablero de juego

18 mazorcas. Clavadas verticalmente en el tablero de juego. 9 de ellas de color blanco, y las otras 9 de color negro. Las mazorcas blancas son fácilmente extraíbles de donde hayan sido clavadas, mientras que las mazorcas negras están atornilladas al suelo, imposibilitando su extracción.

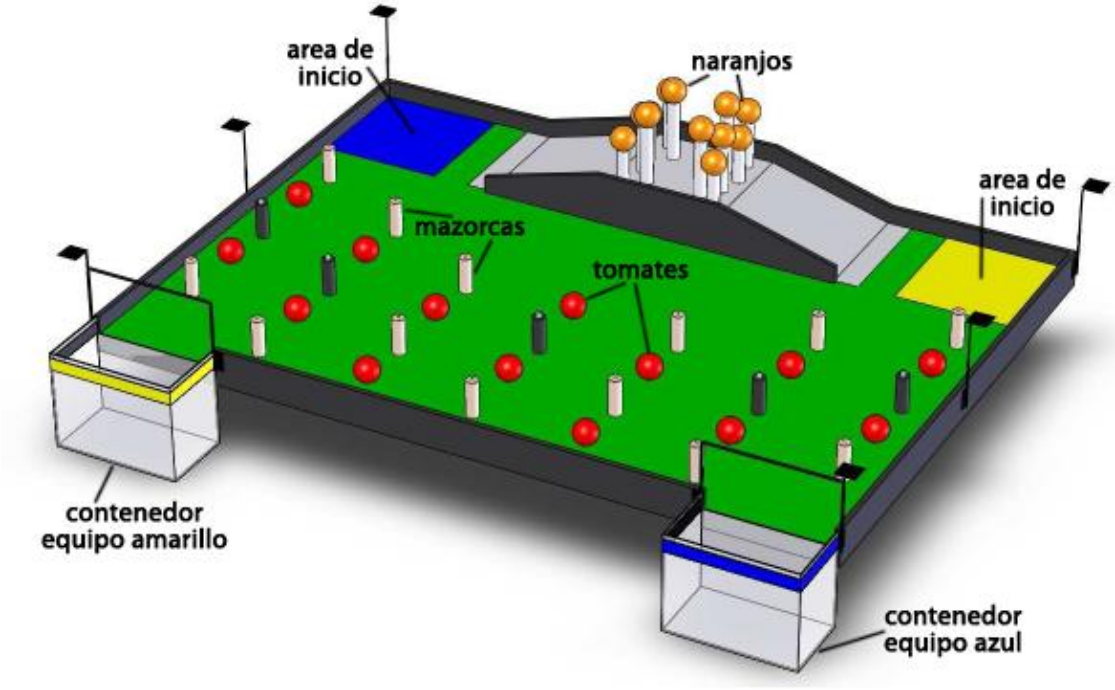

Ilustración 3-11: Campo de juego Eurobot 2010.

Puntuación

Cada equipo debe recolectar la mayor cantidad de alimentos posibles (tomates, mazorcas o naranjas) que se encuentran en los naranjos o sueltas en el tablero de juego y depositarlas en el contenedor que le corresponde, el ***de su color***.

Al terminar los 90 segundos del partido, se hará el recuento en función del peso de los elementos recolectados, de la siguiente forma:

Los ***tomates*** pesan 150g, por lo tanto, cada tomate contenido en el contenedor de un equipo sumará ***150 puntos*** a ese equipo.

Las ***mazorcas*** pesan 250g, por lo que cada mazorca existente en el contenedor de un equipo aumentará en ***250 puntos*** el marcador de ese equipo.

Las ***naranjas*** pesan 300g. Cada naranja contenida en el contenedor de un equipo le dará una cantidad de ***300 puntos***.

Bibliografía

1. **Real Academia Española.** Diccionario de la Real Academia Española. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=rob%C3%B3tica.

2. **Bermejo, Sergi.** *Desarrollo de robots basados en el comportamiento.* s.l. : Ediciones UPC, 2003. págs. 26-27. ISBN 84-8301-712-1.

3. **Technologic Systems.** TS-7350 Linux2.6 FPGA Single Board Computer. [En línea] [Citado el: 12 de 6 de 2010.] http://www.embeddedarm.com/products/board-detail.php?product=TS-7350.

4. **Albillo Arribas, José Ignacio.** Diseño electrónico de un micro-robot Eurobot 2008. *Proyecto Fin de Carrera; José Ignacio Albillo Arribas.* Madrid, España : s.n., 2008.

5. **Linux RT.** Real-Time Linux Wiki. [En línea] [Citado el: 26 de mayo de 2010.] https://rt.wiki.kernel.org/index.php/Main_Page.

6. **Wikipedia.** Modulación por ancho de pulsos. [En línea] [Citado el: 10 de 5 de 2010.] http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos.

7. **Wiki-Robotics.** Cuaderno técnico 3: Servos Futaba 3003. [En línea] 31 de julio de 2003. [Citado el: 8 de abril de 2010.] http://www.iearobotics.com/proyectos/cuadernos/ct3/ct3.html.

8. **Wikipedia.** Odometría - Wikipedia, la enciclopedia libre. [En línea] 14 de marzo de 2010. [Citado el: 12 de mayo de 2010.] http://en.wikipedia.org/wiki/Odometry.

9. **Simulated Reality Systems.** Simulated Reality Systems, LLC - Odometry. [En línea] 15 de enero de 2007. [Citado el: 27 de mayo de 2010.] http://www.simreal.com/content/Odometry.

10. **Eclipse Foundation.** About Eclipse Foundation. [En línea] [Citado el: 20 de 6 de 2010.] http://www.eclipse.org/org/.

11. **Technologic Systems.** FTP Technologic System. [En línea] [Citado el: 18 de 5 de 2010.] ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7350-linux/binaries/ts-images/README .